Biblioteca Para El Diseño

Decarga de libros para

Arquitectura, Diseño Grafico, Diseño Industrial y Artístico

en General.

Formato PDF

Completamente Gratis

designbooksme.blogspot.com

WORKING THROUGH SCREENS

100 ideas for envisioning powerful, engaging, and productive user experiences in knowledge work

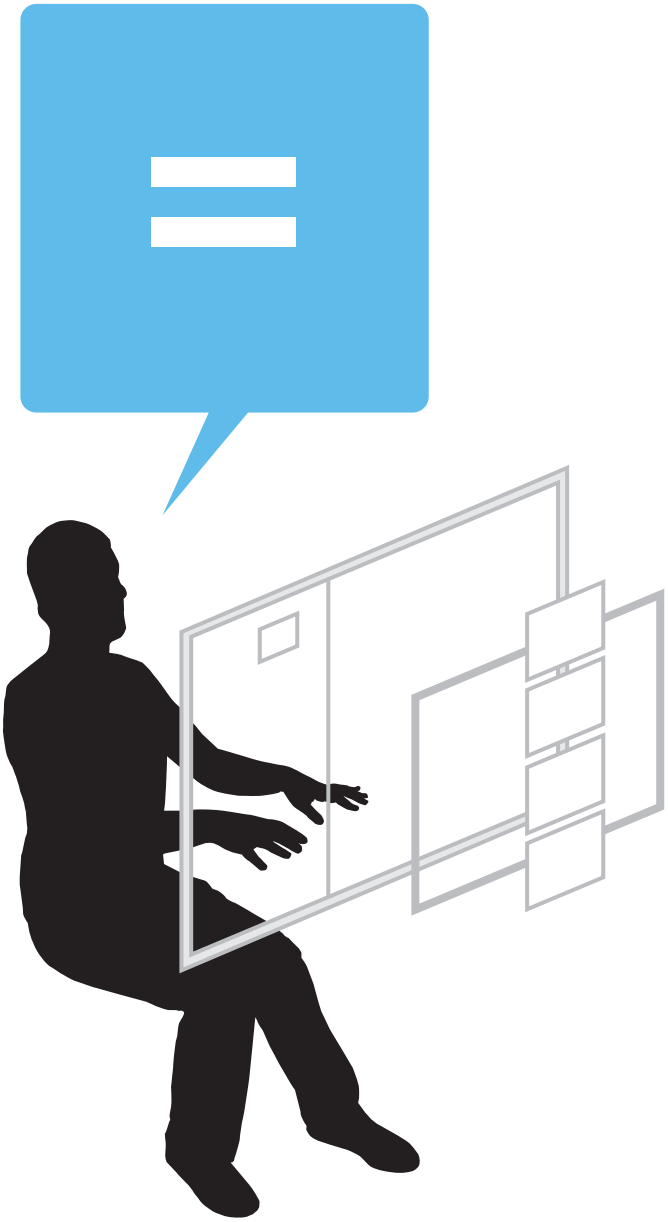
By Jacob Burghardt

Application Concepting Series
No. 1

A publication of
FLASHBULB INTERACTION, Inc

Also available in .html, “Idea Cards” and 11”X17” .pdf formats at
www.FlashbulbInteraction.com

This book is for my grandfather, William Wolfram, who believed that the nature of work was changing into something very different than what he had experienced at sea, in the fields, and on assembly lines — and strongly encouraged me to explore what it might mean.



The category of human efforts sometimes called “knowledge work” is growing.

Knowledge workers are valued for their specialized intellectual skills and their ability to act on and with complex information in goal oriented ways.

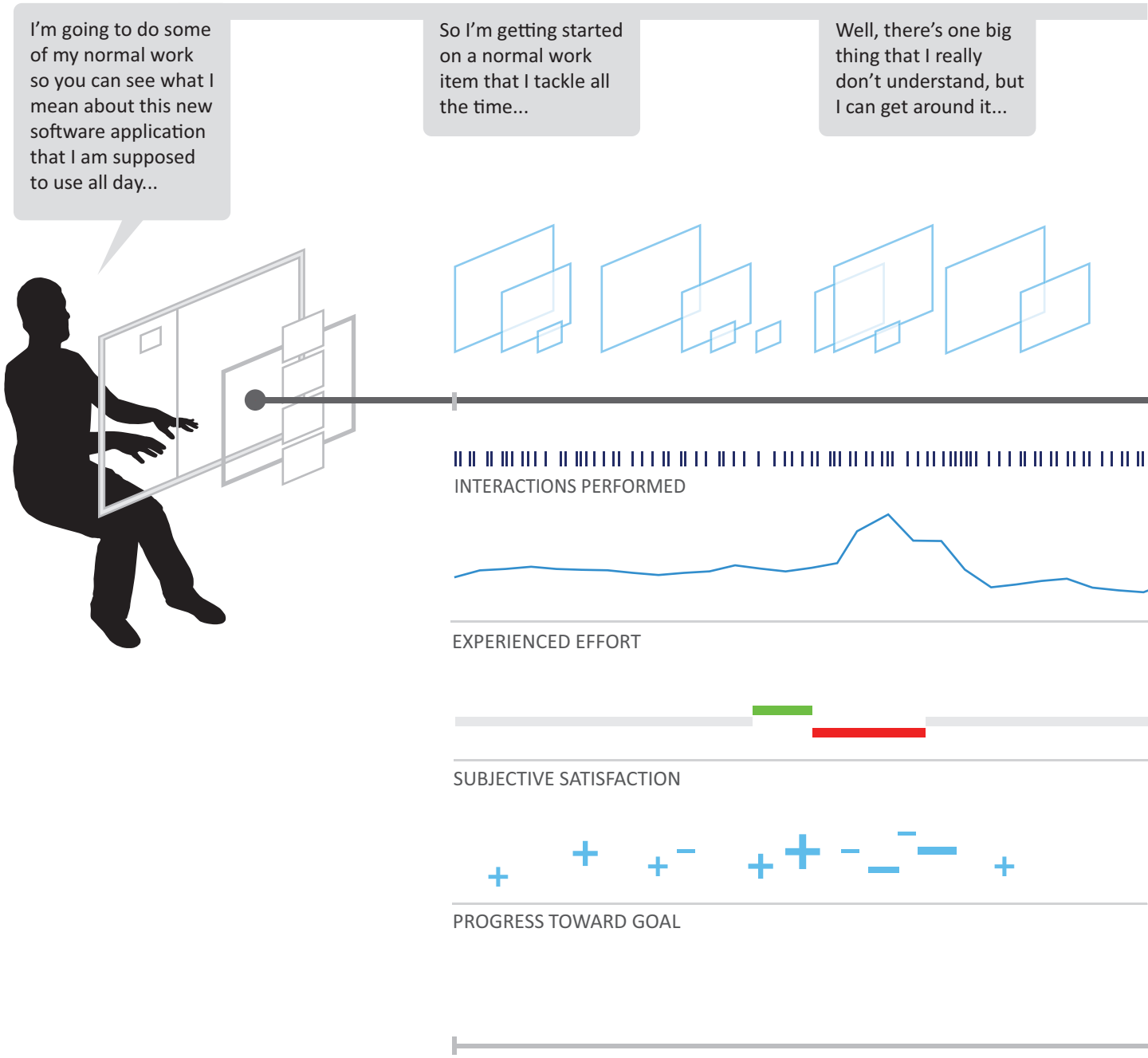
In many contexts, the idea of knowledge work has become almost synonymous with using a computer, to both positive and negative effect.

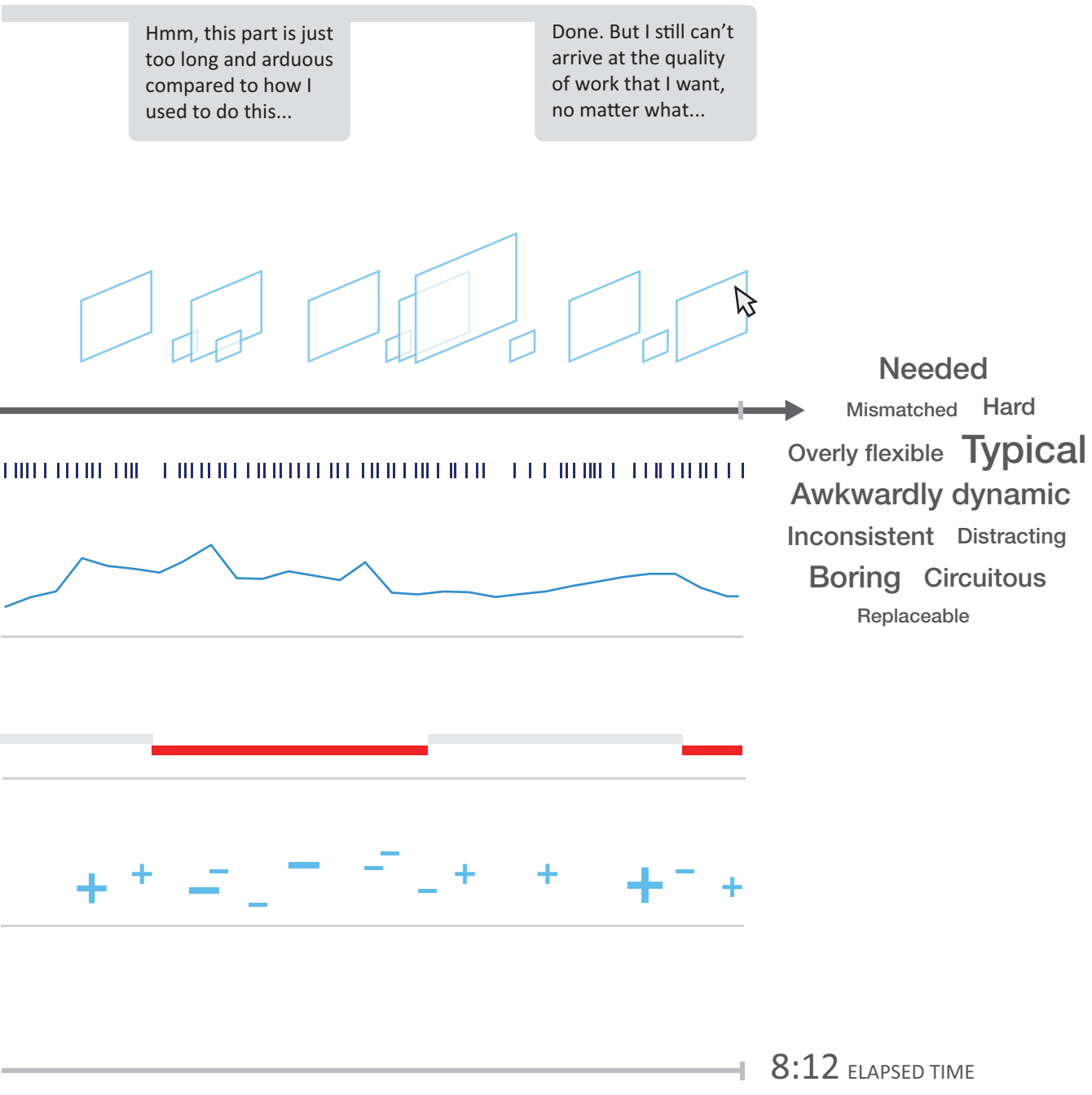
Product teams creating computing tools for specialized workers struggle to understand what is needed and to successfully satisfy a myriad of constraints.

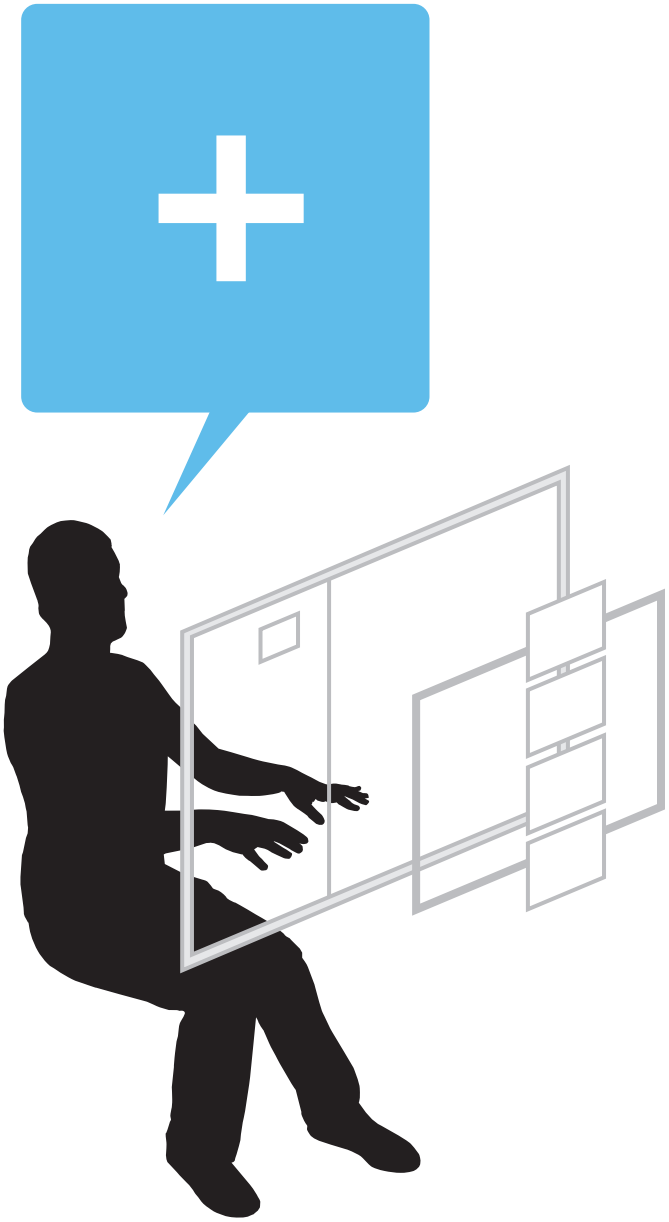
As a result of the design deficiencies in these interactive products, people experience many frustrations in their working lives.

Noticeable deficiencies, along with the ones that have invisibly become the status quo, can lower the quality and quantity of workers’ desired outputs.

With so many people in front of so many screens — attempting to practice their chosen professions — these deficiencies have real costs.


I'm going to do some of my normal work so you can see what I mean about this new software application that I am supposed to use all day...
So I'm getting started on a normal work item that I tackle all the time...
Well, there's one big thing that I really don't understand, but I can get around it...
INTERACTIONS PERFORMED
EXPERIENCED EFFORT
SUBJECTIVE SATISFACTION
PROGRESS TOWARD GOAL


Hmm, this part is just too long and arduous compared to how I used to do this...
Done. But I still can't arrive at the quality of work that I want, no matter what...
Needed
Mismatched
Hard
Overly flexible
Typical
Awkwardly dynamic
Inconsistent
Distracting
Boring
Circuitous
Replaceable
8:12 ELAPSED TIME



Collectively, we have an infrastructural sense of what these technologies can be that tends to limit our ability to imagine better offerings.

Targeted improvements in the design of these tools can have large impacts on workers' experiences. Visionary design can advance entire fields and industries.

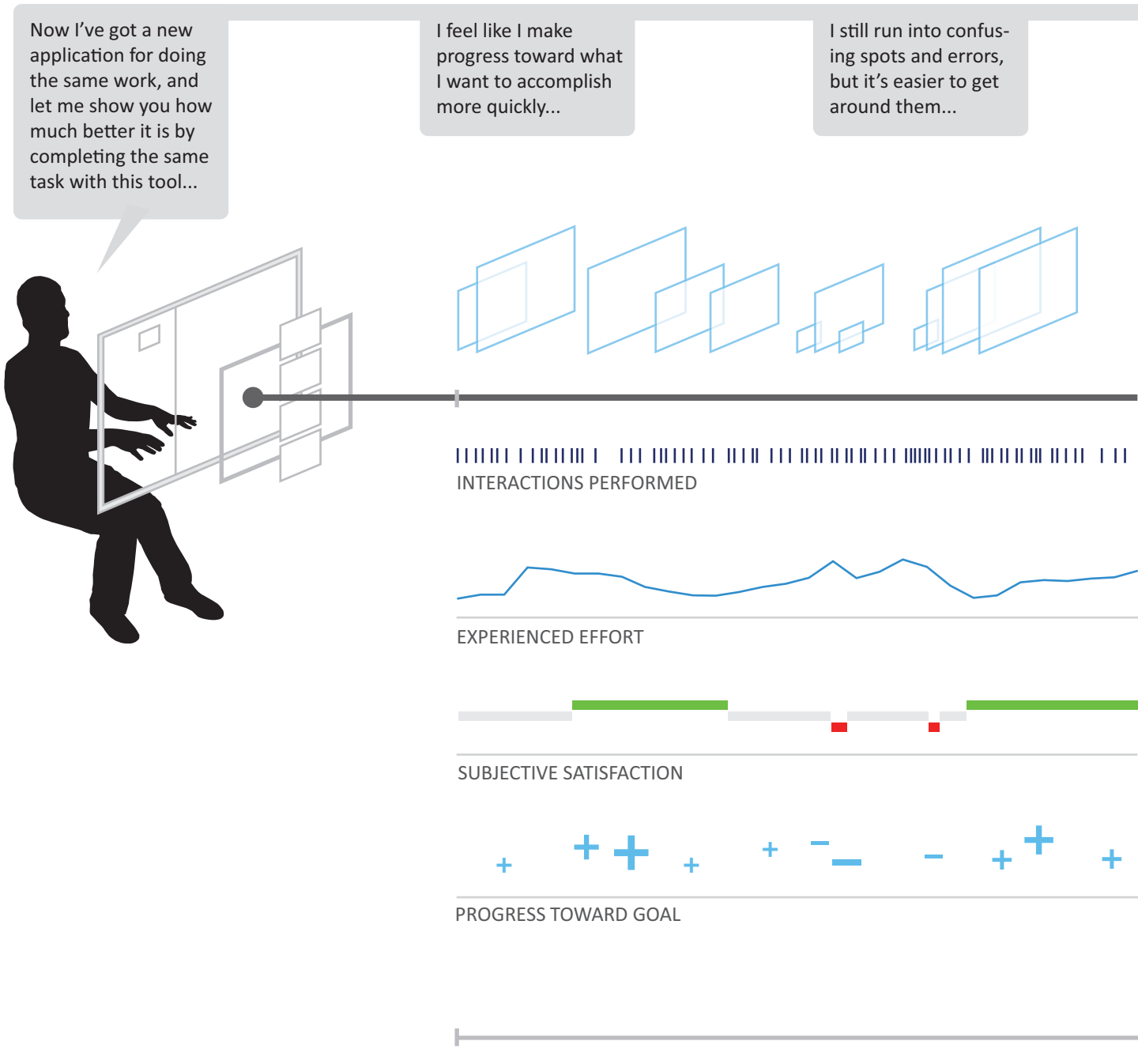
At a basic level, applications can "fit" the working cultures that they are designed for, rather than forcing unwanted changes in established activities. They can augment rather than redefine.

When workers alter their culture to adopt a new computing tool, it can be solely because that tool provides new meaning and value in their practices.

Going further, elegantly designed applications can become a joy to use, providing an empowering, connective sense of direct action and a pleasing sensory environment for people to think "within."

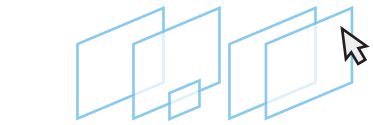
Product teams can make significant progress toward these aims by changing how they get started on designing their products — by beginning with an emphasis on getting to the right design strategy and design concepts long before getting to the right design details.

It is time to start holistically envisioning exemplary new tools for thought that target valuable intersections of work activity and technological possibility.


Now I've got a new application for doing the same work, and let me show you how much better it is by completing the same task with this tool...
I feel like I make progress toward what I want to accomplish more quickly...
I still run into confusing spots and errors, but it's easier to get around them...
INTERACTIONS PERFORMED
EXPERIENCED EFFORT
SUBJECTIVE SATISFACTION
PROGRESS TOWARD GOAL

And I get to a better conclusion faster, which feels much more empowering...

Overall, this new tool feels like it just belongs in how I think about my own ways of working...



Wanted

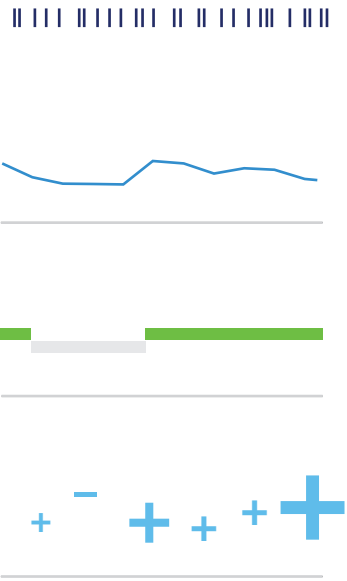
Meaningful Engaging

Clearly targeted **Extraordinary**

Eye opening Dependable activity infrastructure

Domain grounded Mastery building

Irreplaceable **Beautiful**



6:03 ELAPSED TIME

Extensive concepting,

based on intensive questioning,

driving visionary, collaboratively defined strategies

for exemplary tools for thought.

Suggestions for product teams:

Deliberately spend more time envisioning, at a high level, what your interactive application could be and how it could become valued infrastructure in work activities.

Do not assume that a compelling knowledge work tool will arise solely from the iterative aggregation of many discrete decisions during the long haul of a product development process.

Create a divergent ecosystem of concepts for your product's big picture and primary experiences.

Examine the potential value of reusing expected design conventions — while at the same time ideating potential departures and differentiated offerings.

Explore a breadth of directions and strategies before choosing a course.

Plan on staying true to the big ideas imbedded in the concepts that your team selects, while knowing that those ideas will evolve along the way to becoming a reality.

Extensive concepting,

based on intensive questioning,

driving visionary, collaboratively defined strategies

for exemplary tools for thought.

Suggestions for product teams:

Ask more envisioning questions, both within your team and within your targeted markets.

Develop empathy for knowledge workers by going into the field to inform your notions of what your product could become.

Stimulate conversations with this book and other sources relevant to the topic of mediating knowledge work with technology.

Find and explore situations that are analogous to the work practices that your team is targeting.

Keep asking questions until you uncover driving factors that resonate.

Create visual models of them.

Focus your team on these shared kernels of understanding and insight.

Lay the groundwork for inspiration.

Extensive concepting,

based on intensive questioning,

driving visionary, collaboratively defined strategies

for exemplary tools for thought.

Suggestions for product teams:

Use design thinking to expand upon and transform your product's high level mandates and strategy.

Continually explore the strategic implications of your team's most inspiring ideas about mediating knowledge work.

Make projections and connections in the context of key trends and today's realities.

Think end to end, as if your product was a service, either literally or in spirit.

Build and extend brands based on the user experiences that your team is striving to make possible — and how your product will deliver on those promises.

Envision what knowledge workers want and need but do not articulate when confronted with a blank canvas or a legacy of unsatisfactory tools.

Invite workers to be your collaborators, maintaining a healthy level of humility in the face of their expertise.

Extensive concepting,

based on intensive questioning,

driving visionary, collaboratively defined strategies

for exemplary tools for thought.

Suggestions for product teams:

Dive into the specific cognitive challenges of knowledge workers' practices in order to uncover new sources of product meaning and value.

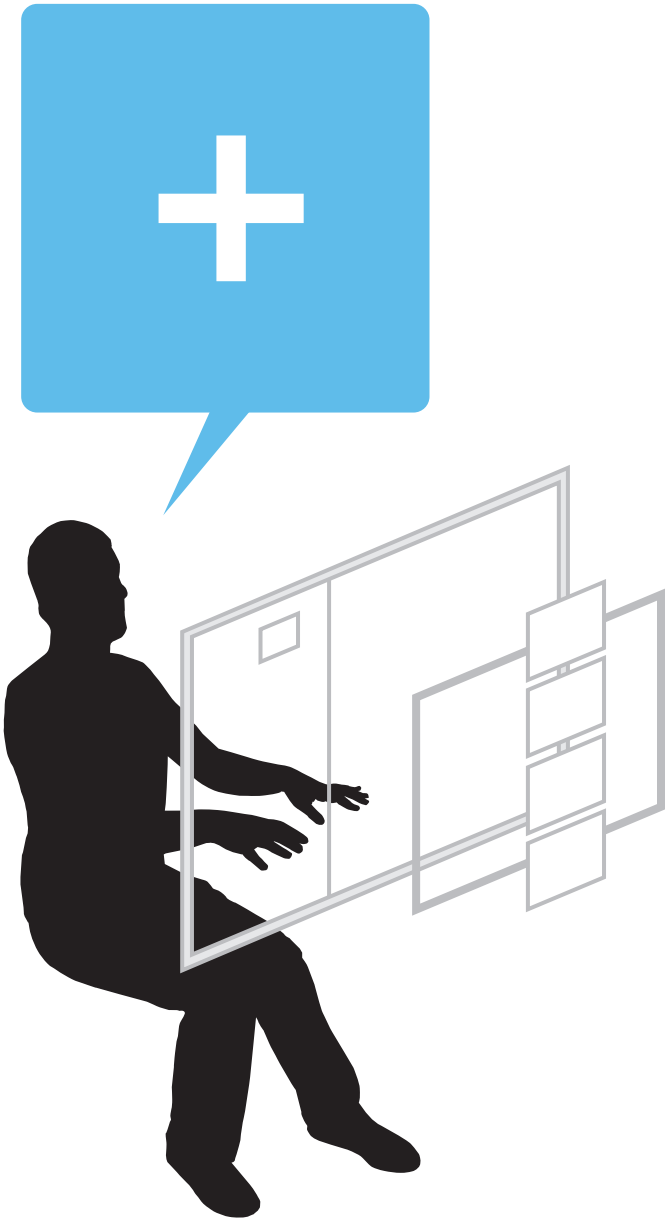
Set higher goals for users' experiences.

Envision "flashbulb interactions" in targeted activities — augmenting interactions that could make complex conclusions clear or open new vistas of thought.

Explore how carefully designed stimuli and behaviors within onscreen tools might promote emotional responses that are conducive to attentive, focused thinking.

Surpass workers' expectations for the potential role of computing in their mental lives.

Raise the bar in your targeted markets, and with it, the bar for all knowledge work tools.



Extensive concepting, based on intensive questioning, driving visionary, collaboratively defined strategies for exemplary tools for thought.

This phrase embodies a suggested overall approach for product teams envisioning new or improved interactive applications for knowledge work.

In support of this suggested approach, this book contains 100 ideas — along with many examples and questions — to help product teams generate design strategies and design concepts that could become useful, meaningful, and valuable onscreen offerings.

Table of Contents

Preface

When I started the writing that eventually resulted in this book, I was driven by a conviction that some critical conversations seemed to be missing from the development of new technologies for knowledge workers.

I kept returning to the same four observations about how many real world product teams operate:

1. Many product teams overlook common needs that knowledge workers have of their onscreen tools while at the same time developing unneeded functionality. These teams start with a seemingly blank slate, even when many valuable product requirements could be explored based on existing, proven understandings of how computing tools can valuably support knowledge work.

2. Many product teams' everyday yet pivotal definition and design conversations do not sufficiently consider knowledge workers' thought processes or how a technology might influence them. While individuals in these teams may occasionally use terminology borrowed from cognitive psychology, the actual details of how a tool could meaningfully impact "thinking work" may not receive more than a surface examination.

3. Many product teams struggle to understand the knowledge work that they are striving to support. Even when some of a team's members have a strong empathy for targeted work practices, teams as a whole can have mixed levels of success meaningfully translating their cumulative understanding into overall models of how their tool could valuably mediate certain activities. These shared models, when executed well, can guide the definition and development of a product's many particulars. Without them, resulting applications can become direct reflections of a team's lack of guiding focus.

4. Many product teams begin construction of final products with very limited notions of what their finished product will be. Whether unintentionally or intentionally, based on prevailing ideologies, they do not develop a robust design strategy for their application, let alone consider divergent high level approaches in order to create a compelling application concept. Instead, they seem to assume that useful, usable, and desirable products arise solely from the iterative sum of many small definition, design, and implementation decisions.

These observations would not carry much weight if it was not for the current state of computing tools that are available to knowledge workers in many vocations. Put simply, these products often contain vast room for improvement, especially in highly specialized forms of work, where there are concrete opportunities to truly tailor technologies to important activities. Highly trained individuals, working in their chosen professions, commonly spend unnecessary effort acting "on" and "around" poorly conceived tools, rather than "through" them. The toll on performance and work outcomes resulting from these extra efforts can be drastic to individual workers, but since it is difficult to collectively recognize and quantify, the aggregate of these losses remains largely undetected within organizations, professions, industries, and economies.

I believe that current deficiencies in technologies for knowledge work are strongly tied to our often low expectations of what it can mean to support complicated activities with computing. Our shared ideas of what constitutes innovation in this space have, in many cases, become tightly constrained by our infrastructural sense of what these technologies can and should be. Too often, we are not seeing the proverbial forest due to our shared focus on a small grove of trees. In our cultural accommodation to what computing has come to "mean" in our working lives, it seems that we may have lost some of our capacity for visionary thinking.

To regain this vision, product teams can spend more time considering what it might actually take to support and build upon knowledge workers' skills and abilities. Getting inside of these essential problems can require teams to adopt goals that are more like those of the pioneers of interactive

computing, who were driven by the potential for augmenting human capabilities with new technologies. When teams extend these pioneering ideas by applying them at the intersection of specific activities and working cultures, they can discover a similar spirit of considered inquiry and exploration.

Higher order goals — aimed at creating tools for thought to be used in targeted work practices, cooperative contexts, and technological environments — can lead product teams to ask very different questions than those that they currently explore during early product development. Through the critical lens of these elevated goals, the four observations listed above can truly take on the appearance of lost opportunities for innovation and product success.

I have personally experienced these lost opportunities in my own career researching and designing knowledge work tools for domains such as life science, financial trading, and graphic design, among others. Even with the best intentions, in 20/20 hindsight, I did not always have time to think through and apply some important ideas — ideas that could have improved products' design strategies and, in the end, enhanced workers' user experiences. There are simply so many useful ideas for these complex, multifaceted problems, and under the demands of real world product development, time for questioning and exploration nearly always passes too quickly.

Listening to other practitioners in the field, I know that I am not alone in making these observations and facing these challenges. And yet, when it comes to accessible, practitioner oriented references on these topics, there seems to be large areas of empty space waiting to be filled.

This book is a foray into part of that empty space. The 100 ideas contained within can act as shared probes for product teams to use in formative discussions that set the overall direction and priorities of new or iteratively improved applications for thinking work. As a collection, these ideas present a supporting framework for teams striving to see past unsatisfactory, "business as usual" technologies in order to create compelling and meaningful tools for knowledge workers at the forefronts of their fields.

I look forward to hearing about how these ideas hold up in the context of your own product development challenges. My sincere hope is that this book provides some measure of inspiration that leads you to envision tools that promote more powerful, engaging, and productive user experiences. Knowledge workers — those who will opportunistically make use of the fruits of your efforts, if you are fortunate — deserve no less.

Jacob Burghardt
1 Nov 2008, Seattle, WA
E - info@FlashbulbInteraction.com
P - 206.280.3135

Acknowledgements

Since this book feels more like a synthesis with a particular perspective than a completely original work, I would like to emphatically thank the authors of all the publications that are included in the bibliography. I would particularly like to thank William Lidwell, Katrina Holden, and Jill Butler — the authors of *Universal Principles of Design: 100 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design* — which was a key inspiration for the format of this work.

The following reviewers have provided invaluable comments on various drafts of this publication: Liberty Harrington, Kristina Voros, Amii LaPointe, Myer Harrell, Aaron Louie, Brian Kuan Wood, Jessica Burghardt, Matt Carthum, Matt Turpin, Miles Hunter, Julianne Bryant, Eric Klein, Chris Ziobro, Jon Fukuda, and Judy Ramey.

I would also like to thank understanding friends who spend long, internally motivated, solitary hours working on personal pursuits. You made this project seem not only possible, but like a good idea.

Publication Information

Working through Screens is the inaugural publication of FLASHBULB INTERACTION, Inc.

This book is available for free in .html and .pdf at www.FlashbulbInteraction.com, where you can also find an abbreviated “Idea Cards” version designed for use in product ideation exercises.

Softcover copies of this book can be purchased at minimum third party cost at: http://stores.lulu.com/flashbulbinteraction

All original contents of this publication are subject to the Creative Commons license (Attribution-Non-Commercial-ShareAlike http://creativecommons.org/licenses/by-nc-sa/3.0/) unless otherwise noted. Please attribute the work to: “Jacob Burghardt / FLASHBULB INTERACTION Consultancy.”

Introduction: The Case for *Application Envisioning*

The Experience of Modern Knowledge Work

In a growing number of contemporary workplaces, people are valued for their specialized intellectual skills and their ability to act on and with complex information in goal oriented ways. There is a general sense that many types of work are becoming more abstract, specialized, complex, improvisational, and cerebral.

Peter Drucker called the people that engage in these types of work "Knowledge Workers." Robert Reich, the former U.S. Labor Secretary, used the term "Symbolic Analysts" to describe a similar category within the workforce. More recently, Richard Florida has defined the characteristics of "the Creative Class." All three of these terms fall within roughly the same frame, emphasizing the commonality of inventing, producing, interpreting, manipulating, transforming, applying, and communicating information as principle preoccupations of these workers.

The current experience of this purportedly new work — what it feels like to practice a highly trained profession or to simply earn a paycheck — has a very different essential character than the type of work experiences that were available just a generation or two ago. A large part of that change in character is due to the extensive use of computing tools in these work practices.

In essence, the expansion of "knowledge work" as a concept has been closely tied to the expansion of computing. Interactive applications have become woven into the fabric of vast territories of professional activity, and workers are continuously adopting new tools into previously "offline" areas. Although these tools are not the only focal point for knowledge workers, they are becoming a point of increasing gravity as cultures of practice continue to co-evolve with these technologies over time.

Consider these example experiences, which are part of the working lives of three fictional knowledge workers who will appear throughout this book:

An architect considers an alternate placement for an interior wall in order to improve the view corridors within a building that she is designing. As she interactively visualizes a certain wall placement within a 3D model of the building, she pauses to consider its implications for a number of the project's requirements. She saves different versions of her design exploration, adding working notes on what she thinks of each design direction. Once she has created several different directions, she then uses the building modeling application to realistically render each possibility, compare them in sequence, and review a subset of design options with her colleagues.

A scientist sorts through the results of a recent clinical study using an analysis application that automatically generates clear and manipulable visualizations of large data sets. She uses the tool to visually locate interesting trends in the clinical results, narrowing in on unusual categories of data at progressively deeper levels of detail. To better understand certain selections within the complex biological information, she downloads related reference content from up to date research repositories.

A financial trader works through transaction after transaction, examining graphs of key variables and triggering his trading application to automatically accept other trades with similar characteristics. He uses his market information application to analyze trends so that he can make better decisions about uncertain and questionable deals. As he barrels through as much work as possible during his always too short trading day, he values how his tools prevent him from making crucial errors while permitting him to act rapidly and to great effect.

While these short descriptions are probably not representative of your own day to day activities, it may be easy enough for you to imagine how essential interactive applications could become in each of these cases. After long periods of accommoda-

tion, accomplishing many knowledge work goals involves turning to a screen, controlling a cursor, entering data, and interacting with well known and meaningful representations of information. Looking toward future technologies, it is likely that most knowledge workers will perform at least some of their efforts within the bounds of a similar framework for some time to come.

The Impacts of Application Design

The design of these computing tools has the potential to make massive impacts on working lives. Unless knowledge workers are highly motivated early adopters that are willing and able to make use of most anything, their experiences as users of interactive applications can vary drastically. These differences in experience can largely depend on the overall alignment of an individual's intentions and understandings with the specifics of a tool's design. Since the majority of the computing applications in use at the time of writing were not created by the workers that use them, this means that the product teams developing these applications contribute roughly half of this essential alignment between user and computing artifact. To restate this common premise, "outside" technologists (of the stripe that would likely be drawn to reading this book) often set the stage for initial success or failure in workers' experiences of their onscreen tools.

Direct alignment with an augmenting tool can cause surprising joy, or at least a sort of transparent, "on to the next thing" sense of success. Individuals and organizations can place a high value on useful and usable products that support workers' limitations while at the same time enhancing their skills. Truly successful interactive applications can provide users with tailored functionality that, among other things, facilitates and enhances certain work practices, powerfully removes unwanted effort through automation, and generates dynamic displays that make complex relationships clear.

In short, when interactive applications are at their thoughtfully envisioned best, they can become seemingly indispensable in knowledge work. At their most visionary, these tools can promote user experiences that provide a sense of mastery and direct engagement, the feeling of working through the screen on information and interactive objects that become the almost palpable subjects of users' intentions.

Issues in Contemporary Onscreen Tools

Unfortunately, many knowledge work products present themselves as nowhere near their thoughtfully envisioned best. Workers too often find that many parts of their specialized computing tools are not useful or usable in the context of their own goals or the larger systems of cultural meaning and activity that surround them. Problematic applications can continuously present workers with confusing and frustrating barriers that they must traverse in order to generate useful outcomes. At their poorly envisioned worst, computing tools can — contrary to marketing claims of advanced utility — effectively deskill users by preventing them from acting in ways that even remotely resemble their preferred practices. Not exactly the brand promise that anyone has in mind when they start the ball rolling on a new technology.

If one was to summarize the status quo, it might sound something like this: when it comes to interactive applications for knowledge work, products that are considered essential are not always satisfactory. In fact, they may be deeply flawed in ways that we commonly do not recognize given our current expectations of these tools. With our collective sights set low, we overlook many faults.

Poorly envisioned knowledge work applications can:

- **Attempt to drive types of work onto the screen that are not conducive to being mediated** by interactive computing as we know

it today. New applications and functionalities are not always the answer, and some work practices can be more effectively accomplished outside of the confines of a computer.

Fail to reflect essential divisions of how work is segmented within targeted organizations, forcing unwanted redefinition of individuals' roles and responsibilities and creating new opportunities for day to day errors in workers' practices.

Introduce new work processes that standardize activities in unwelcome ways. When technologies inappropriately enforce strict workflow and cumbersome interaction constraints, these tools can force knowledge workers to create and repeatedly enact unnecessarily effortful workarounds in order to reach desired outcomes.

Lack clear conceptual models of what they, as tools, are intended to do, how they essentially work, and how they can provide value. Inarticulate or counter intuitive conceptual models, which often stem from a product team's own confusion about what they are creating, can lead workers to develop alternate conceptions of application processes. These alternate models may in turn lead to seemingly undiagnosable errors and underutilized functionality.

Present workers with confusing data structures and representations of information that do not correlate to the artifacts that they are used to thinking about in their own work practices. To effectively use an application built upon unfamiliar abstractions, workers must repeatedly translate their own domain expertise to match a system's definitions.

Encourage a sense of information overload by allowing individuals and organizations to create and store large volumes of valuable information without providing them sufficient means to organize, visualize, navigate, search or otherwise make use of it.

Disrupt workers' attentions, and the essential cognitive flow of intensive thinking work, with unnecessary content and distracting messaging.

Require workers to waste effort entering specifics and "jumping through hoops" that neither they nor their organizations perceive as necessary.

Force workers to excessively translate their goals into the constraints of onscreen interaction, even after extended use. All applications require their users to act within the boundaries of their functional options, but certain constraints on basic actions may be too restrictive and cumbersome.

Introduce automation that actually makes work more effortful, rather than less. Without appropriate visibility into an automated routine's processing, workers can be left with the difficult challenge of trying to understand what has been automated, if and where problems have occurred, and how to fix important issues.

Hide useful historical cues about how content came to be in its current state, while preventing workers from restoring certain information to its earlier incarnations. Tools without these capabilities can increase the difficulty of recovering from errors, which can in turn reduce creativity and scenario oriented thinking in dynamic interactions.

Leave workers without sufficient cues about the activities of their colleagues. This lack of awareness can lead to misunderstandings, duplicated effort, and the need to extensively coordinate efforts outside the computing tool itself. These negative effects may be found in intrinsically collaborative work as well as efforts that are not typically recognized as having cooperative aspects.

Fail to support informal communication

in the contexts where knowledge work is accomplished, as well as provide direct means for actively initiating conversations about key outputs. These omissions can make essential communication acts more effortful, as workers attempt to create common ground and tie their ideas back into application content while using separate, "outside" communication channels.

Lack needed connectivity options for individuals and organizations to tie the product's data and functionalities into their broader technology environments. Resulting applications can become isolated "islands" that may require considerable extra effort in order to meaningfully incorporate their capabilities and outputs into important work activities.

These example points, which represent just a sampling of the many problems that can be found in poorly envisioned knowledge work applications, call attention to the fact that these potential issues in users' experiences are not "soft" considerations. All of these points have implications for workers' satisfaction with a computing tool, their discretionary use of it, the quantity and quality of their work outcomes, and their perceptions of a product's brand. The sum of the above points can be viewed as a fundamental threat to the core goals of organizations that are seeking to adopt new technologies as a means of supporting their knowledge workforces.

Making Do with the Status Quo

Since many of today's applications contain a mixture of both clear and direct functional options and functionality that is frustrating, obtuse, and effectively useless, knowledge workers often become skilled at identifying those portions of technologies that demonstrate benefits relevant to their challenges. Individuals tend to weed out problematic features from their practices, while at the same time salvaging tried and true methods. Over time, the plasticity of mind and culture can display a remarkable ability to overcome barriers and interweave "satisficed" benefits. After considerable effort, established work arounds and narrow, well worn paths of interaction can emerge. An uncompelling, difficult tool can become another necessary reality. The status quo continues, despite the ongoing promise of augmenting specialized, thinking work with computing.

At the level of individual knowledge workers' experiences, attempting to adopt and use poorly conceived applications can lead to frustration, anxiety and fatigue. These negative mental states are not conducive to people successfully accomplishing their goals or being satisfied in their working lives. Put another way, knowledge work applications have the capacity to detract from the pleasure and well being that people experience as part of working in their chosen professions. Knowledge workers often do not contribute their efforts solely for compensation in an economic sense; their actions are intertwined with personal purpose and identity. For this reason, a major deficiency in a knowledge work application can be said to have a different essential quality than a failure in, for example, an entertainment technology. When a knowledge work application becomes an obstruction in its users' practices, vital time and effort is wasted. Beyond the obvious business implications of such obstructions, it is difficult to sufficiently underscore the potential importance of these losses to individual workers, especially when developing products for highly skilled individuals who are seeking to make their chosen contributions to society and the world.

So how did we get here? Where did this status quo come from? Why are these tools not better designed? Why do the brand names of so many knowledge work products conjure disdain, or only a vague sense of comfort after having been extensively used — instead of something more extraordinary? We can assume that no product team sets out to deliver a poorly conceived tool to knowledge workers. And yet, even with good inten-

tions, that is what many have done and continue to do. Ironically, even tools designed for niche, domain specific markets — which can represent the most concrete opportunities to create truly refined tools for specific work practices — are not immune to these problems. In fact, they may be especially susceptible to them.

First Steps of Application Design

Taking a step back, it can be useful to examine the early, initiating steps that lead to the creation of a knowledge work application. Plans for a new or revised computing tool can arise in a variety of ways, though there are some common patterns to their early gestations. In general, a small core of initiators defines a product's principle mandates before a broader cross section of team members and disciplines are brought onto a project. These early conversations may take on very different forms depending on, for example, whether a product represents a disruptive technology or a competitive entry into an established category of knowledge work tools. In any case, teams' invest some part of their formative discussions considering their offerings' potential driving forces, brand positioning, and underlying technological characteristics. These efforts typically involve modeling ideas about potential opportunities in targeted market segments, which often correspond to a particular range of knowledge work specialties and organization types.

During this early initiation, product strategy efforts for knowledge work applications often do not involve "design thinking" in any real sense. When faced with the complexities of scoping and conceiving a viable computing tool, design ideation, at the time of writing, seems to typically take a back seat role. This is in stark contrast to many other types of products, especially outside of computing, where design thinking is increasingly being used as a key approach in early, initiating conversations. One does not need to look very far to see how generative concepting of potential user experiences has become a central exercise in the development of many of today's successful brands and product strategies. Yet in the much "younger" and relatively distant disciplines that develop complex onscreen applications, the potential for design's strategic contributions has not been adequately recognized.

Getting to Design Details Too Quickly

At the end of a knowledge work product's initiating conversations, when it appears that a project will become a funded and staffed reality, there is often a strong desire from all involved to see "something" other than high level abstraction and textual description. The common response to this desire is where foundational user experience problems begin to crystallize. In a characteristic *straight to the details progression*, teams quickly, instinctually move from high level consideration of product strategy into the smallest specifics of a product's definition, design, and implementation. Their approach jumps abruptly from the global to the extremely granular, without the connective tissue of a holistic middle ground.

Part of the reason for this jump in collective mindset is an increase in team size. Left to their own devices, newly added team members often gravitate toward the level of granularity that is their primary focus during the extended course of product development. To a specialist, this makes perfect sense. These detailed skills are what they are typically valued and promoted for, and their narrow expert perspectives are presumably why they are brought onto projects in the first place. The problem with these assumptions is that, when getting into details too soon and too narrowly, specialists' decisions may be under informed and lacking a larger vector of creativity and guiding constraints.

The commonly cited maxim of the influential designer Charles Eames, "the details are not the details, the details make the design," is a useful

truism in the extended development of viable computing applications for knowledge work. After all, if a specific part of a user interface is missing important options for the work practices that a tool is designed for, then its usefulness and usability will suffer during real world interactions. Armed with this understanding, some technologists immediately begin their journey away from the vagaries of a product's strategy toward something more "real." Without considering how they might be stifling their own success and innovations, these teams begin haphazardly anticipating workers' detailed needs and possible complaints as a means of sketching a satisfactory concept for their product.

The path of the *straight to the details progression* is predictable and common. Product teams enacting this progression begin implementing without the vector of a larger design strategy to guide them through the many highly specific choices that will inevitably follow. Their initial conception of their product is relatively simplistic, but they believe that they can continually map out the complex specifics along the way, whether in diagrammatic illustrations, textual specifications, or in working code. They move forward with the implicit assumption that interactive applications, being made of abstract computer language, are somehow highly malleable, and that all encompassing "fixes" can be made when needed.

In reality, product teams creating knowledge work applications rarely have the luxury of extensive downstream revisions, despite their deep seated assumptions to the contrary. When they do enjoy the luxury of such changes, the cost of these revisions can be prohibitively high. For this reason, key corrections, additions, and improvements are all too often put off for the "next version," or "next public release" with the assumption that users will be able to work their way around any issues in the meantime. Facing limited resources and complex challenges, many teams develop distorted notions of what constitutes acceptable, or even exceptional, quality and user experience.

While specifying every detail of a complex interactive application before any implementation takes place is also not generally considered a viable approach to product development, at the time of writing, the pendulum seems to have swung too far in the direction of improvising design strategy. Prevailing straight to the details ideologies are largely out of step with the reality of resulting product outcomes. A survey of the inflexibilities, over extended interaction frameworks, and scattered conceptual models of contemporary knowledge work products in many domains can sufficiently prove this point.

Adding Features Until "Magic Happens"

Behind the *straight to the details progression* is a belief that a successful, even visionary, product will somehow emerge from the sum of countless detailed definition, design, and implementation decisions (see Figure 1 on page 34). In this view, applications can evolve from a collection of somewhat modular pieces, so long as the assemblage does not somehow "break" in the context of users' human limitations and cultural expectations. Keep working on the details and magic will happen — or so the assumption goes.

The larger gestalt of an interactive application receives little or no consideration in this framing of product development. Teams with this mindset do not typically sketch diverse concepts for how their creation could mediate work practice in appropriate, innovative, and valuable ways. To overstate the case, many product teams believe that knowledge workers can be supported by directly giving them what they want, adding details to a tool as needed in a somewhat systematic manner. This approach may work for a while — until tools collapses along fundamental, structural fault lines of conceptual clarity, information display, and meaningful consistency.

Even though the *magic happens expectation* often results in poorly designed computing tools for knowledge work, the *straight to the details progression* may be successfully applied to other types of onscreen products. This might explain why many product teams creating knowledge work applications still hold on to these shared assumptions — there are positive examples and well known brand names that can serve as their reference points. When a product's goals are relatively simple or very well characterized, as in a highly established genre of application, teams can have a shared grounding without actively taking time to grow that collective understanding. For example, everyone in a typical product team probably understands how a collaborative calendar application works, because they use them every day. If their understanding happens to be less than complete, team members can probably round out their views without too much difficulty or discussion. A product team may even be able to create real innovations in this kind of application by making incremental changes in small details based on assumptions about unmet needs.

Crucial Understanding Gaps

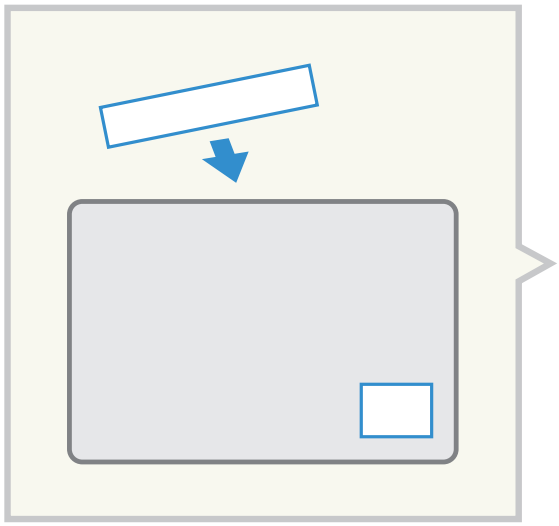
Tools for specialized knowledge work typically do not fit this sort of "make it up as we go" mold. One of the main reasons is that product teams inevitably have a difficult time understanding the work practices that they are striving to mediate. They do not tacitly know the cultures that they are attempting to support. A base level of understanding about larger systems of activity and meaning is necessary in order to design a useful tool that will be well suited for those systems. Teams need to understand what the architect Eliel Saarinen spoke of as the "next larger context." Software developers, for example, do not inherently know what it means to analyze clinical research data, let alone how that data fits into the larger flows of activity within a research lab.

When technologists find it difficult to understand the many specifics of foreign and elaborate work practices, they may unwittingly hold onto an initial, roughly hewn, consensus view about knowledge workers' activities and needs. This view can become their framing point of reference throughout the development of their product, despite incoming information that could valuably transform it. In practice, the momentum of a disoriented group's initial concept for their computing tool often places certain ideas at the primary, driving core of what is eventually developed and released. What the architect and psychologist Bryan Lawson calls a "primary driver" takes hold in their design outputs. And in these cases, as end users of such products can attest, magic does not often happen.

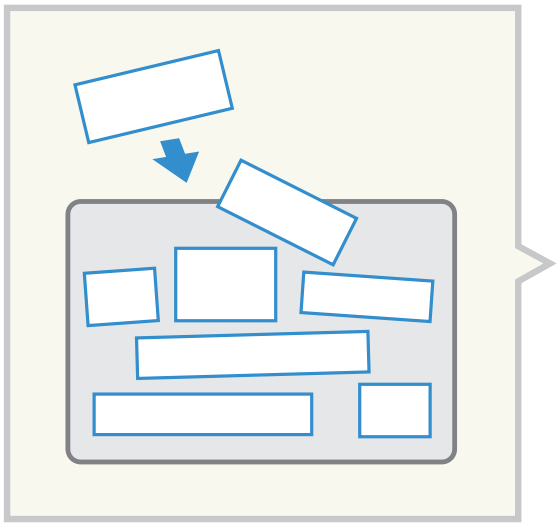
Uncritical Reliance on Pioneering Ideas

If the pioneers of interactive computing had only been thinking about detailed design decisions, at the expense of the bigger picture, they would have likely never envisioned many of the conventions that we commonly use today. For example, Douglas Englebart, a pivotal figure in the pioneering era, has defined much of his working life based on a series of epiphanies about how technology could enhance human problem solving. During a time when computers were still primarily used for batch process mathematical tasks, he envisioned remarkable possibilities for the application of computing to knowledge work. Of particular interest is Englebart's astonishing 1962 description of an architect using interactive computing as a fluid part of complex work practices, long before such a future had been realized. In his essay "Augmenting Human Intellect: A Conceptual Framework," Englebart outlined how an architect might use a computer to review a symbolic representation of a building site; consider different scenarios in excavation and building design; refer to handbook and catalog resources; locate windows so that light is not reflected into the eyes of passing drivers; examine the resulting structure to ensure that it does not contain functional oversights; and store the resulting work for later retrieval and annotation by stakeholders (the architectural examples used

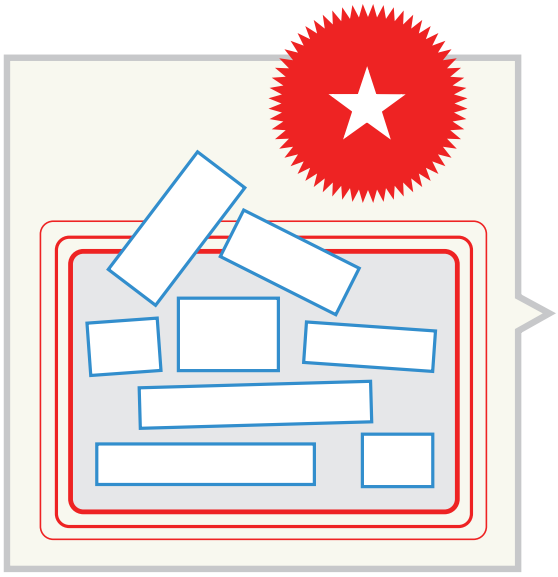
FIGURE 1. COMMON APPROACH TO ITERATIVE APPLICATION DESIGN



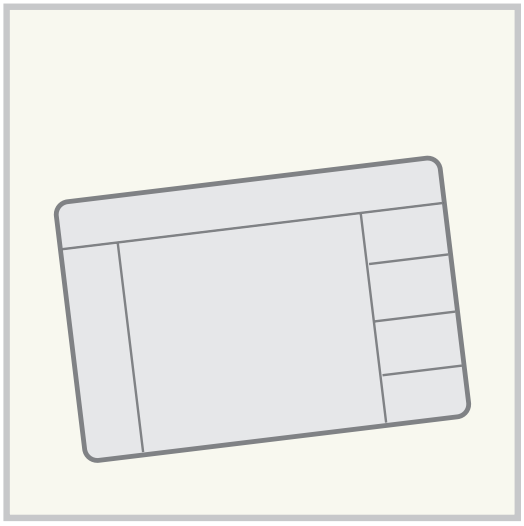
Begin creating individual features, without spending any time in the space between high level product strategy and detailed product implementation



Iteratively add more discrete parts, without considering overarching ideas about how the application could mediate knowledge work



Until magic happens, somehow unifying the aggregation of separately created minutiae



And a cohesive, or at least satisfactory, application supposedly emerges

In reality, such products may be deeply and frustratingly flawed, driving poor user experience and lesser outcomes in targeted knowledge work

throughout this book are an homage to Englebart's landmark application concept).

Pioneers of interactive computing, such as Englebart, did not have the luxury of working only at the detailed level of their emerging creations. They also set the vision and goals for their own and subsequent generations of technological development. Looking objectively at the conversations taking place in product teams today, it appears that many technologists are relying very heavily on these and other proceeding foundations. Not on the intellectual spirit of these foundations, but on their literal conventions. As knowledge work applications have become standardized and commonplace within technologists' worldviews, it seems that we may have all become limited by a shared, infrastructural sense of what these tools can and should be. People creating these products have, to some extent, stopped examining them through a critical lens that could uncover important new possibilities. As they continue to copy and tweak existing standards, we become increasingly accustomed to a certain rate of change and a certain level of generic, all purpose design.

While vernacular evolution certainly has its place, repetition of familiar patterns is clearly not the entire picture of exceptional design process. Knowledge work tools can be much more than the sum of their discrete functional parts. A sole focus on detailed salvaging and assembling of the past leaves no room for other, important pursuits. If product teams do not explore different strategies for their application's overall approach to mediating work, how will they imagine new tools that truly and valuably fit into workers' specialized thought processes and cultures?

Embracing a More Strategic Creativity

Appropriate and exciting concepts for knowledge work tools are built on holistic vision, not just pattern matching and incremental iteration. They require a carefully considered design strategy to tame their potential complexities into clear, useful, and desirable simplicity.

The very idea of design strategy implies the selection of one direction from a pool of potential approaches, yet the *magic happens expectation* restrains breadth and ideation by promoting a narrow track of implemented reality. In essence, teams following the *straight to the details progression* are practicing *single vision and concept design*. The essential, elemental "shapes" of their products are the shapes that happen to unfold in front of them after the sum of many small decisions. They deemphasize a larger type creativity, which in turn reduces possibilities for useful and compelling innovation.

So how can product teams creating interactive applications for knowledge work embrace this larger type of creativity? If the *straight to the details progression*, the *magic happens expectation*, and *single vision and concept design* characterize the mindset that eventually leads to problematic or failed computing tools, what mindset can teams adopt to avoid these pitfalls?

Introducing *Application Envisioning*

Generally speaking, product teams can cultivate a perspective of targeted yet open exploration, without analysis paralysis. They can spend more time in the space between product origination and product implementation. They can create an environment where divergence and a multiplicity of ideas are valued in their discussions. They can forgo an early emphasis on specifics by creating abstract models that visualize their understandings and outline potential spaces of design possibility. They can ask more questions in their targeted markets and sketch novel concepts for how their products could play a role in knowledge work, while documenting tangible evidence of their ideas. They can balance top down decision making with bottom up input from knowledge workers in order

to synthesize singular design strategies. These strategies can embody a strong brand positioning and the grounding of a team's best application concept, assembled from a core set of sketched functionalities that target a carefully chosen scope of work practices.

This suggested approach can be summarized by the following phrase, which appears in the opening pages of this book:

> Extensive concepting, based on intensive questioning, driving visionary, collaboratively defined strategies for examplary tools for thought.

Is there a repeatable methodology or process to advance this change in mindset and general approach? Not in any strict sense, because these explorations are very emergent and freeform, despite their focused nature. However, a name for this period between project initiation and project implementation could allow teams to effectively plan for it. The term *application envisioning* suggests an early, separate interval in product development in which teams can intentionally and collaboratively consider potential design strategies and design concepts for their computing tool, rather than sliding down a largely unconsidered course (see Figure 2 on the next page).

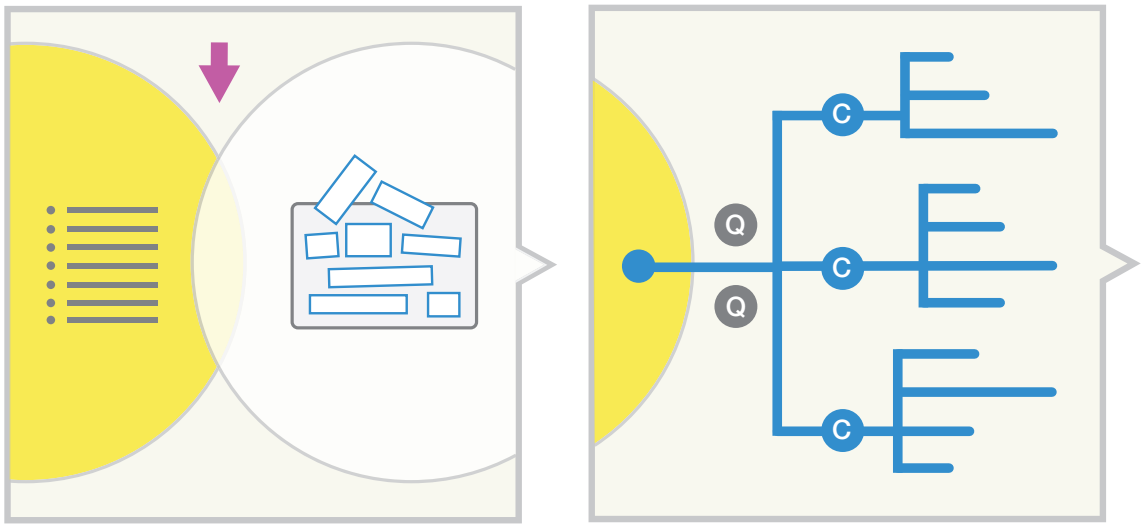
Application envisioning can allow teams to cultivate empathy for targeted knowledge workers and their worlds, lay the ground work for inspiration, explore diverse questions and ideas about what their product could be, and develop a shared, big picture view — with the assumption that many important details will need to be fleshed out along the way to a completed release.

One (increasingly routine) process suggestion for *application envisioning* is that this early, explorative time presents a significant opportunity for product teams to get out of their offices and into the field. Teams can strive for "what it's like" understanding of knowledge workers' current experiences by directly observing and engaging in their worlds. While immersed in the activities that they are striving to mediate with computing, teams can uncover unmet needs and other important insights for design strategy. This immersion may also lead them to start thinking about their product as a service, either literally or in spirit, which can highlight new areas for innovation through ongoing, networked connection. Teams may take a sense of partnership with targeted workers so far as to invite them to become collaborators, maintaining a healthy level of humility in the face of their expertise.

Another process suggestion is for product teams to look outside of the work that they are targeting in order to cast new light on their envisioning questions and their emerging design concepts. While pioneering figures of interactive computing had to work from an essentially blank slate, today's technologists do not have to start from square one when they think about what it might mean to augment certain thought processes and activities with computing. There is a growing body of research and critical perspective that teams can use as lenses for making sense of these complex, multifaceted design problems. In order to extract potential strategic principles, teams can examine computing tools that have been successfully adopted into similar activity contexts within other types of work practice. Advanced analogies to products in other domains can lead to inspiration that may fuel truly novel solutions that draw upon seemingly unrelated fields of endeavor.

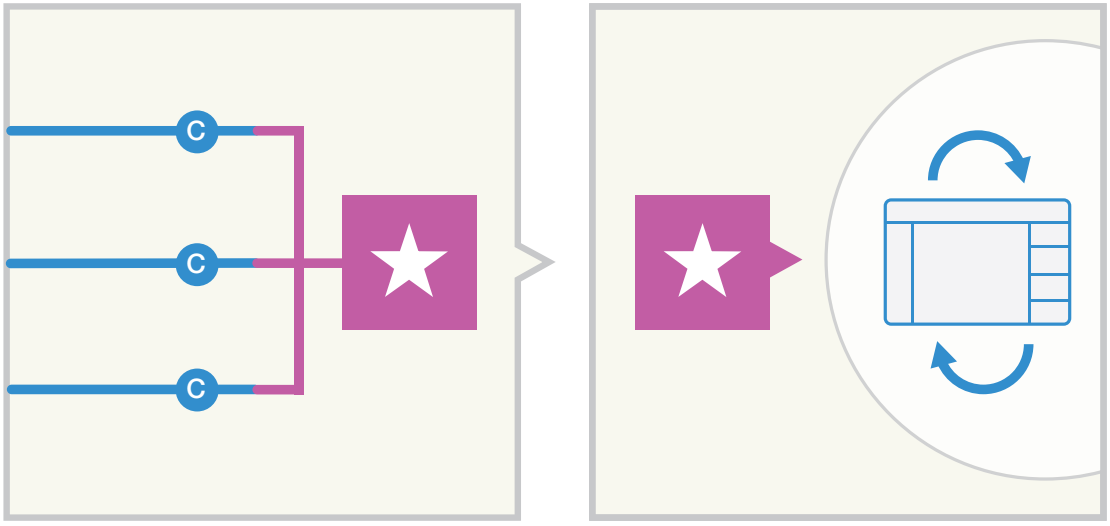
The idea of *application envisioning* has strong parallels to mindsets found in other, older design disciplines, whose practitioners more commonly apply design thinking in strategic ways. For example, product teams creating computing tools for knowledge work can learn a great deal about envisioning new technologies from the successful practices of the best industrial design teams. These teams also shape peoples' daily lives through their creations, albeit with a focus on the mass produced, physical embodiment of material culture. Industrial designers typically take time early in their projects to explore different concepts so that they

FIGURE 2. *APPLICATION ENVISIONING* APPROACH TO DESIGN



Spend more time in the space between high level product strategy and detailed product implementation

Meaningfully question what it could mean to mediate certain knowledge work activities with technology: observing and talking with targeted workers, collaboratively modeling the problem space, and sketching diverse design concepts



Strategically synthesize the fittest overarching vision and concept for your product from among an ecosystem of envisioned futures

Then move forward with your chosen design strategy and design concepts, expanding upon details, iteratively implementing and gathering further input

can divine the "right" overarching direction for their product, rather than immediately honing in on and elaborating a single solution. These designers often conduct various forms of research, synthesizing models of their problem space before moving forward into design ideation. Once they begin ideating, they typically sketch thumbnail after thumbnail of potential options, long before they even consider realistic renderings or exacting specifications. From these early explorations in "design research," industrial design teams can uncover important constraints, possibilities, and languages for their product. They can discover potential emotional connections with end users and gain empathy for the context of a successful offering and brand, all of which puts them in strong position to define singular and compelling design strategies.

The Higher Goals of "Flashbulb Interactions"

Envisioning a diverse range of appropriate possibilities for a product is not an easy task. Even with a shared emphasis on a multiplicity of ideas, practitioners of all design disciplines sometimes face the lure of literal, small scale iteration of known patterns when more innovative responses could be appropriate, valuable, and feasible. *Application envisioning* efforts can represent a fundamental change in how product teams define and design interactive applications, but this change alone may not be enough to arrive at exceptional tools for knowledge workers. Without higher order goals that aim to truly augment peoples' intellectual skills and abilities, *application envisioning* can become just another phase in product development, without any of the intended, strategic payoffs. A team's own infrastructural grounding in the conventions of computing can easily stifle threads of divergent, meaningful concepting. The gravity of the known can easily preclude more creative questions and proposals.

A new term may be useful to product teams as they attempt to uncover new sources of value in knowledge work computing. Flashbulb interactions are a branch of sorts off of the term "flashbulb memories," coined in 1977 by Roger Brown and James Kulik in the psychology literature. A flashbulb memory is a recollection that stands out as a clear and pivotal moment, a punctuated experience in the compilation of one's past. In a similar vein, a flashbulb interaction is one of those rare moments when an interactive application impacts a knowledge worker in some profoundly positive way, such as making a complex conclusion clear or opening up a new vista of thought.

Product teams can explore how their computing tools might promote flashbulb interactions by beginning their projects with these high level questions:

> What are the **big picture problems that knowledge workers currently face** in their work practices? What mental work is currently difficult?
>
> How might our application **transform abstract and taxing mental work** into dynamic, highly visual, direct, and appealing interactions?
>
> How could our interactive application help knowledge workers accomplish the **best work of their professional lives**? What would those outcomes look like?
>
> How could our application support highly valued work **outcomes that could not be attained without** its functionalities?
>
> How could our application r**educe or eliminate routine tedium** in knowledge workers' experiences, while allowing them to **use their expertise in new and valuable ways**?
>
> How could our application **foster and clarify useful communication and collaboration**?
>
> How could our application promote a sense of **confident power** and uninterrupted, **focused engagement**?
>
> How might the transition to using our ap-

plication be **a pleasurable experience that workers will remember** for years to come, especially when they reflect on how they used to accomplish the same goals?

These questions are a direct attack on low expectations of technologies for knowledge work. They contain an optimism that is similar to pioneering questions that lead to the creation of interactive computing, but they can be applied to the grounded particulars of specific challenges that product teams face today. Most importantly, when technologists have asked these questions, they may find it difficult to fall back on literal, small scale iteration of known design patterns, knowing full well that more innovative responses could be appropriate, valuable, and feasible.

Product teams are not likely to know if and when they have generated design strategies and conceptual sketches that could result in products that meet these aspirations, but that sort of absolute decision making is not the point of conducting these inquiries. Instead, teams can pose these and other questions about flashbulb interactions in order to take their eyes off of the conventional state of knowledge work computing and begin considering potential narratives for exceptionally positive user experience. This change in perspective can uncover surprising ideas and design constraints that, in turn, can help teams to better understand deep seated opportunities that their application might address, as well as what those solutions might look like.

Summary of Case for *Application Envisioning*

To summarize, contemporary computing tools for knowledge work often contain significant design deficiencies — both recognized and overlooked — that detract from people's working lives. Looking beyond the current state of these tools, interactive computing has remarkable potential for improving thinking work. An early emphasis on design strategy and design concepts, not design details, can be crucial for developing truly successful computing tools in this space. Product teams that embrace early envisioning as a central exercise in application development, along with significantly elevated goals for user experiences, can generate appropriate and innovative possibilities for emerging generations of knowledge work tools. By intensely questioning what it could mean to mediate specific thought processes and work practices with an interactive application, these teams can develop tools that deliver more enjoyable and relevant experiences, better work outcomes, improved brand loyalty, and other valuable results.

Using This Book

This book is a tool for product teams to use as they envision new or iteratively improved knowledge work applications. It presents 100 ideas that can remind teams of common factors for the design of extraordinary computing tools, helping them to generate a greater diversity of sketched models, frameworks, and concepts. Each concisely presented envisioning idea is a specific consideration for early, formative conversations about what an application might become. These random access topics are intentionally enmeshed and overlapping, not mutually exclusive. The categorization of the 100 ideas sketches an overall framework and is intended to improve their collective accessibility as an envisioning reference. The resulting collection is a practitioner oriented synthesis that can expand the range of questions that product teams explore as they generate potential design strategies and design concepts — inherently raising their shared expectations for their products' positive impacts on knowledge work.

The 100 ideas themselves can be traced to a range of sources and perspectives in product strategy, human factors, human computer interaction, systems analysis, industrial design, interaction design, information architecture, usability research, computer science, and other professional specialties. Many of the ideas are rooted in commonly cited considerations and guidelines, though they

have been framed here specifically for use while envisioning computing tools for knowledge work. Some of these commonly cited points call out specific functionalities that are currently available in a subset of contemporary products, while others touch upon broader connections to the technological contexts that workers' practice within. This book also borrows liberally from those authors who have put forward ideas that have advanced my own work as a practitioner providing research, strategy, and design services. These publications can be found in the bibliography. Beyond commonly cited ideas and valued references, a number of the 100 ideas can be traced back to specific stories from real world product teams. These envisioning ideas were considered assumptions in some groups of technologists and missing in others groups in a way that pointed to their value.

A variety of audiences may find the 100 ideas in this book useful:

Product managers and other leaders within organizations can use these ideas to promote innovative design strategies and to inspire their teams to set higher goals for product success.

Researchers investigating the characteristics, practices, and potential technological desires of certain populations of knowledge workers can use these ideas to outline a broader range of questions for their studies.

Definers of interactive applications can use these ideas as probes to generate models and stimulate strategic thinking in workshops and other requirements elaboration efforts.

Designers of interactive applications can use these ideas to identify important user experience factors for different activity contexts, to sketch a broader range of design concepts, and to make more informed decisions about design strategy in this space.

Stakeholders and influencers in *application envisioning* can use these ideas to drive product teams toward a broader conversation about what it might mean to valuably augment specific types of knowledge work.

Students may find this survey of factors informative, gaining a sense for the potential breadth of considerations that can influence the design of these computing tools.

Book Approach and Exclusions

Although much of the text is written as if the reader is part of product team designing a new knowledge work application, the same ideas can apply when revising or extending an existing tool. Similarly, the tone — but not the primary information — of this book often reflects the interests of product teams working in commercial contexts. Please note that this book's ideas might be just as applicable to tools created by an open source community or developed internally within knowledge work organizations.

100 ideas is a very round number, and it points to the limitations of this book. Just as there is no set recipe for effective product development, there are many other, equally valid ideas for envisioning interactive applications for knowledge work. The ideas in this book were selected due to their potential impacts in a wide range of *application envisioning* conversations. Many of the ideas represent generally important considerations that are commonly overlooked in contemporary products. That being said, some of the ideas will presumably be much more important for specific product contexts than others. None of the 100 ideas are universals or do-or-die edicts. Please take them or leave them, depending on the situation you find yourself in and your belief in their value.

The reader will find few mentions of specific technologies in this book, other than frequent references to certain genres of networked applications used in architecture, clinical research, and financial

trading. For example, this book does not focus on Web technologies, even though the 100 envisioning ideas could be extensively applied to Web based tools. There are also limited references within the 100 ideas to specific methodologies, other than some general approaches to modeling work practice (the hierarchy of operations, tasks, and larger activities is coarsely adapted from Alexei N. Leontiev's Activity Theory) and interactions (Ben Shneiderman's "Object-Action Interface Model," without its emphasis on direct manipulation). This exclusion of extensive technology and methodology references was intentional. Ideally, product teams using very different technological foundations and methodological approaches will find this book to be useful. In the end, all viable methodologies have some place for determining an application's essential form and direction, regardless of what that particular process box happens to be called. Please insert this book's *application envisioning* ideas there.

Although this book contains ideas for the development of new technologies, it is anything but some attempt at distant futurism. Instead, the focus here is primarily on personal computing applications that could conceivably be in front of the eyes of knowledge workers at the time of writing, given the state of contemporary technologies. The domain specific examples used throughout will reinforce this focus. Although some of the functionalities described in these examples are presumably not available in real world tools (no specific products were referred to during the writing or illustration of this book), they are intended to represent realistic possibilities for interactive computing in the present tense.

Thirteen Categories of Envisioning Ideas

The 100 envisioning ideas are broken into thirteen different categories that form chapters of sorts. While these chapters are suited to random access skimming, some readers may benefit from having first familiarized themselves with key ideas in categories A, B, and C, such as "Interrelations of operation, task, and activity scenarios" or "Intentional and articulated conceptual models," if they are unfamiliar with these notions.

The following brief descriptions of the thirteen idea categories conclude this introductory section:

Category A, "Exploring work mediation and determining scope," contains nine ideas that can help product teams pursue useful understandings of knowledge work practice. These understandings can inform insightful models and design concepting, which can in turn illuminate where an application could provide appropriate and desirable value in workers' experiences. The ideas in this category describe the potential importance of investigating workers' physical and socio-cultural environments; determining tasks and larger activities that are conducive to mediation with computing tools; and supporting specialized needs related to emergent work, collaborative work, and individual, localized practices.

Category B, "Defining interaction objects," contains ten ideas that can help product teams envision clear, understandable onscreen entities for knowledge workers to act on and with in order to accomplish their goals. The ideas in this category highlight the potential importance of interaction objects' definitions, identification, associations, states, flagged variability, ownership, relationships to specific interactions, and templates.

Category C, "Establishing an application framework," contains ten ideas that can help product teams envision consistent, understandable application concepts that envelope and organize various functionalities for mediating work. The ideas in this category highlight the importance of applications' conceptual models, interaction models, differing levels of interaction patterns, navigation pathways, identity tailored views, states, and other overarching, "structural" considerations.

Category D, "Considering workers' attentions," contains seven ideas that can help product teams

envision functionality concepts that effectively account for the strengths, limitations, expectations, and customs associated with workers' attentions. Teams can refer to this section when envisioning how their applications might support users' desires to remain productively focused on their chosen vocations. The ideas in this category highlight the potential importance of tempos of work, expected effort, opportunity costs, distraction, engagement, resuming work, alerts functionality, the development of habit and automaticity, and other attentional considerations.

Category E, "Providing opportunities to offload effort," contains six ideas that can help product teams to envision functionality concepts that could reduce unwanted knowledge work effort while at the same time keeping workers in the seat of control. The ideas in this category highlight the potential importance of offloading memory burdens; automating appropriate operations, tasks, and activities; allowing workers to maintain an internal locus of control; and providing meaningful visibility into the internal workings of automation.

Category F, "Enhancing information representation," contains eleven ideas that can help product teams envision how systems of tailored and interactive information representations could provide value in targeted knowledge work practices. The ideas in this category highlight the potential importance of representational coordination, genre, novelty, relationships, transformation, and interpretation aids, as well as some specific categories of information display.

Category G, "Clarifying central interactions," contains seven ideas that can help product teams successfully envision key interaction scenarios while fleshing out sketches of their central functionality concepts. The ideas in this category highlight the potential importance of interactive narrative, clarity around levels of selection, specific instances of error management and workspace awareness, support for impromptu tangents, presentation of relevant supporting information, and transitioning work outcomes from private to public view.

Category H, "Supporting outcome exploration and cognitive tracing," contains four ideas that can help product teams envision support for knowledge workers' scenario oriented exploration of potential outcomes, as well as historical review of application content. The ideas in this category highlight the potential importance of versioning, undo, action history for interaction objects or functional areas, and private, working annotations.

Category I, "Working with volumes of information," contains seven ideas that can help product teams envision functionality concepts for managing and working with the masses of information that are generated by, and referenced throughout, knowledge work activities. The ideas in this category highlight the potential importance of flexible organizing methods; searching, filtering, and sorting application content; handling uncertain data sets; integrating information sources; providing messaging around content updates; and archiving unused yet valued information.

Category J, "Facilitating communication," contains seven ideas that can help product teams envision appropriate support for both implicit and active communication in knowledge work practices. The ideas in this category highlight the potential importance of integrated communication actions, representational common ground, work handoffs, authorship information, features to facilitate contact between workers, public annotation of interaction objects and functional areas, standardized genres of communications, and printing options that can fit workers' communication needs.

Category K, "Promoting integration into work practice," contains 13 ideas that can help product teams envision application concepts that, beyond branded marketing claims, are intended to unfold as relevant and approachable tools for targeted tasks and larger activities. Teams can also use these ideas to envision extensibility that could allow targeted individuals and organizations to bind new

tools to their existing computing systems and customs. The ideas in this category highlight the potential importance of application localization, introductory experiences, early attributions of usefulness, differing design approaches based on frequency of access, carefully considered user assistance, application interoperability and integration, end user programming, credibility of content and processes, and "at hand" application reliability.

Category L, "Aiming for aesthetic user experiences," contains five ideas that can help product teams envision a more enjoyable, appealing, domain appropriate, recognizable, and potentially unique directions for their applications' aesthetics. The ideas in this category highlight the potential importance of carefully designed knowledge work outputs, meeting or exceeding contemporary aesthetic standards, exploring small but iconic design resemblances to known domain artifacts, pursuing clear illustration content and direct branding, and considering iconoclastic aesthetics directions.

Category M, "Planning connection with use," contains four ideas that can help product teams envision ways to anticipate, learn from, and support the real world use of their computing tools. The ideas in this category highlight the potential importance of having early and iterative conversations with targeted knowledge workers, supporting system champions that could advance product adoption, fostering and learning from application user communities, and considering the potential for unanticipated uses of technological options, long before their implementation has begun.

Primer on Example Knowledge Work Domains

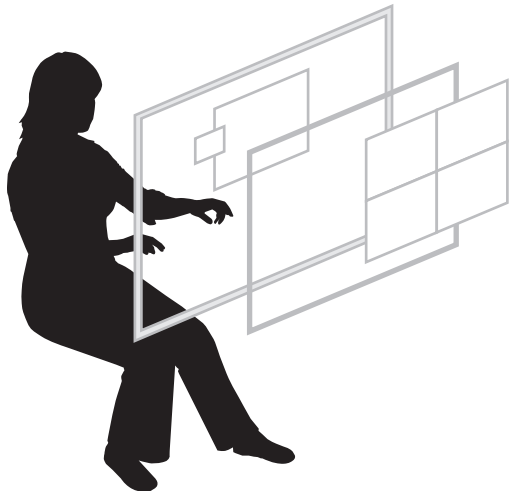
This section contains brief background descriptions of the three knowledge work domains used as examples throughout this book: architecture, clinical research, and financial trading. These example domains show the 100 envisioning ideas "in action" in specific contexts. By including three domains instead of one, each envisioning idea presents an opportunity to illustrate useful parallels and commonalities that can be drawn across very different types of work practice.

The following background content is greatly simplified when compared to the complexity of real work in any one of these three fields. The same can be said for the related examples found throughout the 100 envisioning ideas themselves. Specialists in these professions will likely find this book's descriptions of their vocations to be lacking in important specifics. They are. Please note that these omissions are intentional. This text is a fast access reference to key ideas that can improve *application envisioning* of knowledge work tools, not a comprehensive sourcebook for any one profession.

Architecture

Architects and their firms, generally speaking, seek to profitably create well designed drawings for buildings that address complex criteria. These criteria can be set by diverse stakeholders such as clients, civil engineers, government regulators, and the general public. Architects also set many criteria themselves, based on their training and their personal perspectives on what constitutes good design. To reach these aims, architects frequently transition between synthetic creativity and highly analytical problem solving. The process of arriving at agreed upon building designs, and carrying them forward through construction, can involve many different types of activities and work processes. For this and other reasons, teams of architects and consultants, rather than a single individual, are often responsible for the design of any given project.

Visions of interactive applications in architectural practice began relatively early in the history of computing and continue to hold remarkable promise for future expansion (see the earlier mention of Douglas Englebart's landmark application concept on page 16). These technological possibilities have been tempered by the established professional cultures in many architecture firms, which have historically been relatively slow to adopt available computing tools. At the time of writing, for an important range of reasons that are likely to persist for some time, a considerable amount of architectural practice is still being accomplished outside of computing environments.

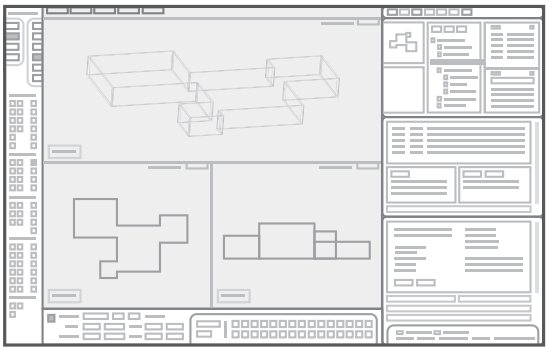


During the intervals of a project where architecture firms do frequently turn to interactive applications, they may use a variety of products, including computer aided drafting (CAD) and other tools for exploring, visualizing, simulating, presenting, revising, detailing, and communicating design possibilities. While some of a firm's applications are usually tailored specifically for architectural practices, architects also employ standard productivity tools and other general purpose products as part of their technological repertoires.

The generations of architects working today have varying desires and expectations for their own use of interactive applications. Some of the more

experienced, senior architects have remained reticent about using computing in tasks that the majority of architects now exclusively accomplish on screen. These experienced professionals often focus on how computing tools can limit the expressiveness and clarity of architectural outputs, while at the same time adding a high degree of learning, abstractness, and complexity to their own work practices. This reticence is in stark contrast to new practitioners in the field, who are expected to have a standard set of skills that includes effective operation of many of the latest computing tools. In between these two extremes are practitioners that are highly skilled at using "their" favored, proven products, and can make these chosen tools fit a wide variety of situations.

At the time of writing, a subset of leading architecture studios has a strong interest in adopting new technologies to accomplish their aims. Some even consider their use of advanced computing applications as one of their key differentiators in the marketplace. Many of the expressive, curvilinear, and asymmetrical geometries found in contemporary architecture would be effectively impossible to resolve without the type of interactive explorations that are available within contemporary computing. Additionally, some cutting edge architects have become interested in how certain tools can programmatically generate novel forms and based on iteratively defined rules and constraints.



Building Information Modeling Application Concept

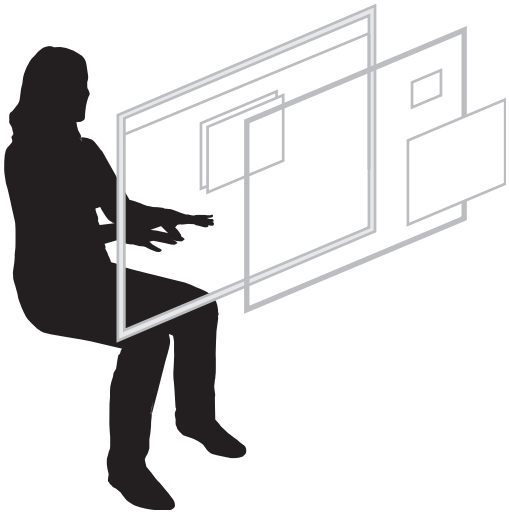
A key, recent development in the industry has been the introduction of Building Information Modeling (BIM), a term that encompasses an emerging class of computing applications that is beginning to drive radical changes in architectural practice. In BIM, the entire design of a building is stored as a collaborative virtual model that can be modified and referenced by different contributors to a project, purportedly improving communication and reducing representational misunderstandings. Since BIM inherently presents many of the challenges that can occur when attempting to support collaborative work with interactive applications, a hypothetical "building modeling application" appears throughout the architectural examples included in the 100 envisioning ideas

The fictional architect in this book's examples works at a medium sized, cutting edge studio with a robust computing infrastructure. She is still in the relatively early phases of her career, though she already has her eye set on becoming a partner some day or starting a similar practice elsewhere. At her level of seniority, she is a generalist, with responsibilities that range from client workshops to iteratively developing design and construction documents. She is part sketchbook dreamer, part diplomat, and part detail oriented workhorse. Her workplace goals include:

- Surpass, or at least meet, client expectations
- Create appealing, functional, high quality designs
- Incorporate compelling ideas and "good design" into building drawings
- Collaborate effectively to meet project budgets and timelines
- Contribute to award winning work that impresses partners in her firm

Clinical Research

Clinical research scientists, generally speaking, want to make applied discoveries related to human health. These scientists adopt diverse methods and technologies to attack their research problems, depending on the nature of the topic under study and researchers' own areas of expertise. Different research questions and methodological approaches are often funded and staffed at different levels, though these levels can change drastically when promising results appear. Ad hoc procedures can quickly become established protocols as a clinical lab's efforts progress from minimally staffed explorations to a larger, production workforce of experimentation.



Life scientists, a larger category to which clinical researchers can be said to belong, were relatively early users of computing, and they have continued to drive some of the most exciting progress in the application of interactive tools to knowledge work. Although time spent at the laboratory bench has remained a staple of many clinical research activities, extensive onscreen work has also become part of the essential character of these scientists' working lives.

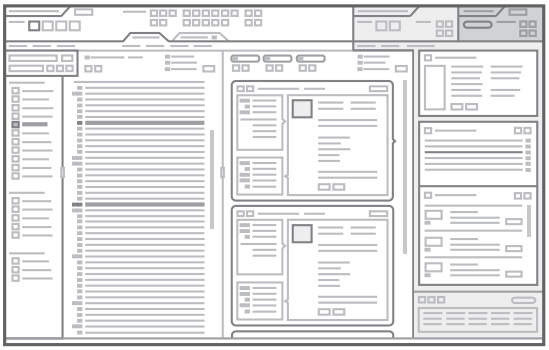
Clinical research labs differ in their adoption of specialized computing tools, based in large part on their budgets and the character of their research. Labs with limited computing infrastructure often focus on storing experimental data in a central repository and providing laboratory staff with typical productivity applications, which they may then supplement with a variety open source tools. At the time of writing, clinical labs with more extensive computing infrastructure have the option to adopt technologies for nearly every stage of experimental workflow, ranging from sample preparation robotics and automated instrumentation, to specialized analysis software for data mining, to electronic laboratory notebooks for keeping track of experimental progress. To the uninitiated, stepping into a large, well funded lab can feel something like stepping into some futuristic version of an industrial production line, with many stations and the buzz of human and machine activity.

Many clinical research labs study the genetic properties of samples in order to understand the presence or absence of characteristics that may be pertinent to their research problems. Making confident conclusions in these types of studies can require a massive number of experiments, resulting in volumes of data that are difficult to manage outside of computing environments.

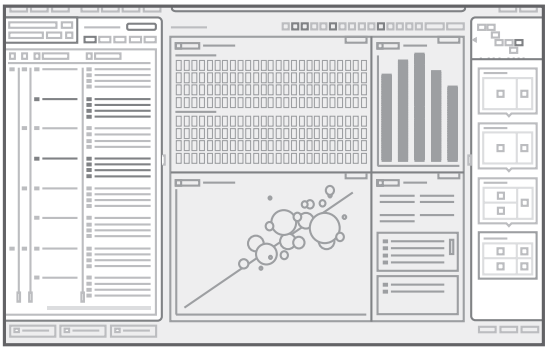
The most frequently used application in many clinical labs is the Laboratory Information Management System (LIMS). LIMS, at its most extensive, keeps track of all stored data about a laboratory, from the stock on the shelves to the results of genetic tests. Many of these systems also provide functionality for defining and monitoring laboratory workflow, allowing scientists to design and distribute experimental protocols for lab technicians and automated instruments to follow. Since LIMS are often open to integration with other applications, they can become a central hub for connecting all of a laboratory's computing infrastructure.

Applications for analyzing clinical data are an important class of technologies that may be connected to a LIMS. The analysis tools designed for the scientific market represent some of the most

advanced examples of interactive applications currently available to knowledge workers. These tools can take seemingly countless pieces of laboratory data and present them in ways that allow scientists to understand trends, uncover anomalies, and make decisions. Robust visualization functionality can allow researchers to sift through experimental results from a variety of perspectives based on emergent wayfinding approaches. In clinical research areas where certain established analyses are often useful for understanding data, highly tailored functions can automate known, well characterized tests and present their results in clear and actionable information displays.



Lab Information Management Application Concept



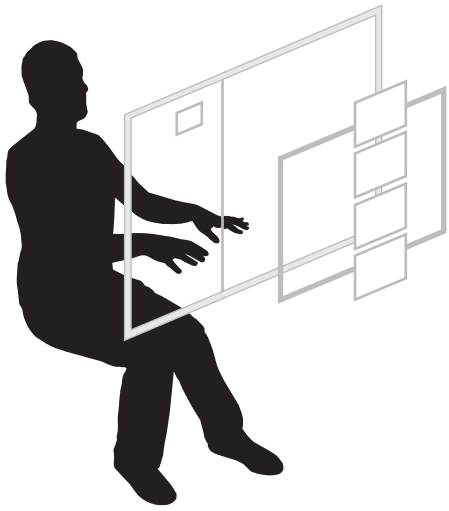
Lab Data Analysis Application Concept

The fictional scientist in this book's examples conducts clinical research, largely funded by government grants, on populations with a deadly hereditary disease. She has had many years of academic training and experience and is valued for her intelligence, depth of knowledge, insights, and personal drive. She has recently become the Principle Investigator of her own research lab, with responsibility over all of its clinical programs and personnel. Her new facility has extensive computing infrastructure, and she has been able to select LIMS and analysis applications that present the best available fit for her planned research approaches. Her workplace goals include:

- Make discoveries that lead to improvements in human health
- Design innovative studies and protocols
- Mentor students and staff
- Ensure that lab technicians have what they need to conduct experiments
- Analyze experimental data as thoroughly as possible
- Publish leading findings in reputable journals
- Manage lab resources wisely

Financial Trading

The many specializations of financial trading are, generally speaking, about the exchange of financial instruments to maximize returns for traders, their firms, and their clients. The teams that accomplish these goals are composed of distinct roles and established hierarchical structures that help ensure strict accountability. One important distinction in financial firms' personnel is the pervasive separation between trading and "back office" groups. While traders make decisions about actions in their markets, the back office completes the detailed work that makes deals happen, such as billing, accounting, and any reconciliation of specifics that might be needed.



The history of financial trading has strong ties to advanced applications of communication technologies. Traders are communicative people, and ongoing relationships based on stable interchanges have traditionally been a necessity in order to secure favorable transactions in markets over time. The desire for the most current market information possible has driven successive generations of traders to rapidly adopt new technologies. For example, one of the first applications of the telegraph was the transmission of market data, and in a similar vein, many financial organizations were relatively early adopters of communication via computer networks.

Computing automation and interactive applications have had profound impacts on professional practice in financial trading. Although contemporary traders may still be vocal participants in their markets, at the time of writing, many types of trading transactions are typically accomplished without any face to face or phone conversation. Instead of verbal interaction, communication in these specialties now commonly involves the exchange of textual information on computer screens. These networked exchanges have created opportunities for trading automation based on predefined, quantitative rules set within and executed by computing tools. In situations where this sort of automation is used extensively, actual conversations outside of one's own firm may occur only in special cases, such as negotiations over large deals, or as an intentional means of building specific business relationships through personal connection.

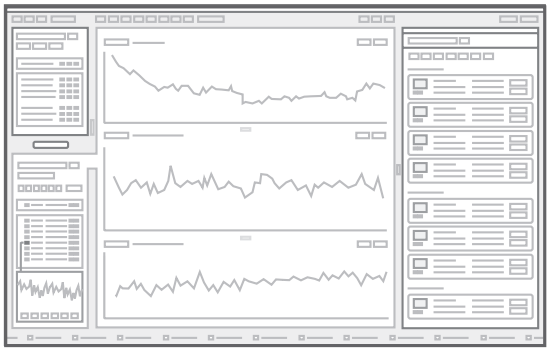
Real time market information feeds, as well as a wealth of online research functionality, have created the potential for information overload and excessive cognitive burdens in traders' work. Successful traders, having adapted to this potentially overwhelming context, become skilled at knowing when to invest time to research a transaction and when it is more beneficial to simply execute a deal based on immediately available information. These choices of time and attention are made, in part, based on the input and visible activities of other traders. Onscreen tools for supporting collaboration are often supplemented with shouts to colleagues across the room or via a global "squawk box" intercom system.

While the use of computing is universal in modern financial organizations, individual firms have varying attitudes about providing new technologies to their workforces. Some firms conduct updates to their computing infrastructure in long, safe cycles, while others are continually attempting to improve the productivity of their staff by providing them

industry leading applications.

The main drivers for adopting new technologies into trading activities have been promised increases in efficiency and volume, reductions in errors, warehousing of useful data, and freeing workers from menial actions so that they can spend more time conducting "smarter" business. Financial firms often develop their own specialized computing tools internally, and when they purchase applications from niche product vendors, they may substantially customize them during their system integration processes. Outside of domain specific products, both traders and back office workers make extensive use of typical, off the shelf productivity applications and communications technologies.

Specialized Trading Application Concept



Specialized Market Analysis Application Concept

The fictional financial trader in this book's examples works in the flagship building of a leading global financial firm. His company is known for making significant investments in computing infrastructure for its highly sought after staff. He has been in financial services for a few years, but is still at a point in his career where he wants to stay focused on day to day trading. He is motivated by monetary rewards, but he also enjoys the responsibility, risk taking, rapid decision making, and intensive, moment to moment focus of market transactions. He is a highly social person, and is known by coworkers and other traders as a wit and conversationalist. His workplace goals include:

- Work fast and smart, making decisions quickly
- Exceed, or at least meet, financial targets
- Maintain business relationships and have good conversations
- Be honest and fair with counterparties while advancing organizational goals
- Keep current on relevant market news and trends

A.

Exploring Work Mediation and Determining Scope

Valued computing tools can seemingly “fit” into certain parts of knowledge workers’ actions and thought processes, usefully meshing within the flows of their own goals.

Designing for such a harmonious pairing requires critical exploration of potential interventions into targeted activities.

During *application envisioning*, product teams can model and rationalize knowledge work from a variety of perspectives in order to understand how certain practices might be usefully mediated by their own onscreen applications.

Teams can use these models to sketch divergent functionality concepts, eventually drafting an appropriate and desirable scope for their computing tool.

Ideas about the potential roles that a product could play in knowledge work can arise in different ways. Product teams working within mature genres can build and innovate based on existing understandings. Teams seeking to create novel applications, whether tailored to a specific workplace or a larger market segment, can have more extensive, "from the ground up" questions to consider. In either situation, teams can intentionally reevaluate and flesh out their initial ideas about their product's contributions to workers' activities.

Since so much of knowledge work is tacit and occurs inside workers minds, it can be difficult for product teams to gather the information that they need to create useful shared models of current work practice and its challenges. Direct observation in work environments and iterative, participatory modeling processes can help teams gain insights into what workers have difficulty remembering and articulating.

Different approaches to modeling work practice can frame certain problem spaces in different ways. Teams can use these differing frames to identify areas for fruitful design concepting, such as needed "basics" for a computing tool, potential areas for improvement, and workers' unmet needs.

This category contains 9 of the 100 *application envisioning* ideas in this book:

A1. Influential physical and cultural environments

A2. Workers' interrelations and relationships

A3. Work practices appropriate for computer mediation

A4. Standardization of work practice through mediation

A5. Interrelations of operation, task, and activity scenarios

A6. Open and emergent work scenarios

A7. Collaboration scenarios and variations

A8. Local practices and scenario variations

A9. High value ratio for targeted work practices

Product teams can use these ideas to explore how different understandings of know-ledge work practice can inform diverse application concepts and refined design strategies. Even when a product's initial charter targets a specific domain goal or activity, more expansive modeling and ideation can highlight opportunities for more systemic responses and valuable innovations.

The central notion of this category applies to all of the *application envisioning* ideas, though it is most closely related to the "Defining interaction objects" (B), "Establishing an application framework" (C), "Promoting integration into work practice" (K), and "Planning connection with use" (M) categories.

A1. Influential Physical and Cultural Environments

The environments that knowledge workers practice within — which includes both their multidisciplinary organizations and the larger cultural context of their professions — can pose key challenges and opportunities for product teams as they attempt to outline appropriate and compelling design strategies.

Examples from three knowledge work domains:

> A financial trader depends on many elements of his office environment to accomplish his work. From the "yelling distance" proximity of key colleagues, to the availability of specialized computing and communication tools, to the in house services that allow him to work late, he feels that his firm has done everything it can to support him as he strives to sit at his desk and focus on maximizing profits for his group **(see illustration on next page)**.
>
> A scientist organizes the spatial layout and bench assignments of her clinical lab to promote frequent, unplanned communication and the effective execution of structured research work. There are few "offices," and most of the computing workstations are placed on or near benches where technicians run experiments.
>
> An architect's desktop computer is situated in an open floor plan room dedicated to a single building project. The walls of the space are covered with large printouts of current work. She typically does not have to go very far to have an informal conversation with anyone on her project team — though she still finds the group to be too hierarchical.

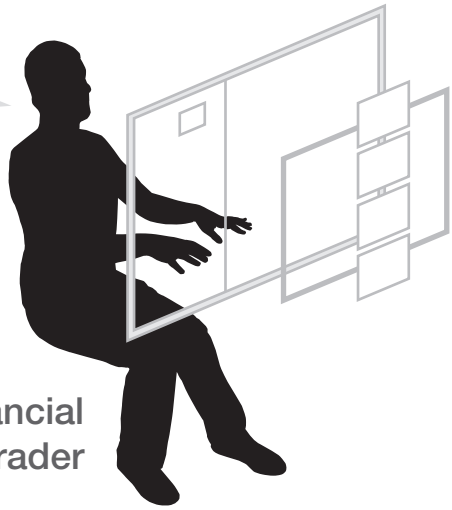
All knowledge work occurs in a physical and cultural environment, and successful individuals can be quite adept at making use of their situational contexts. While the conventional cubicle row remains a stereotyped landscape for knowledge work, many professions have specialized workplace schemes that have evolved throughout their history (C7, G4). Changing organizational structures and philosophies, in conjunction with the expansion of computer networks and other communication technologies (J), have created opportunities for some types of knowledge work to become geographically distributed, "remote" or even "nomadic."

Product teams can holistically model targeted settings in search of valuable insights that could be meaningfully reflected in their divergent application concepts. For example, knowledge workers' immediate cultures can exert powerful influences over the purpose and character of what they consider to be standard norms and customary practices. At a macro level, individual workers may also learn from and contribute to communities of practice that span multiple organizations and geographic locations (M3).

When product teams do not actively consider the potential influence of physical and cultural environment on their emerging ideas about work mediation and application scope, opportunities to clearly situate products within their eventual contexts can be lost. Applications that do not adequately reflect physical realities (K1) and cultural settings (A2, C5, B7) can be more difficult for workers to learn (D2, D3, K2, K6) and may not be seen as useful or attractive options (K3).

See also: A, B8, C4, F2, G7, K10, M

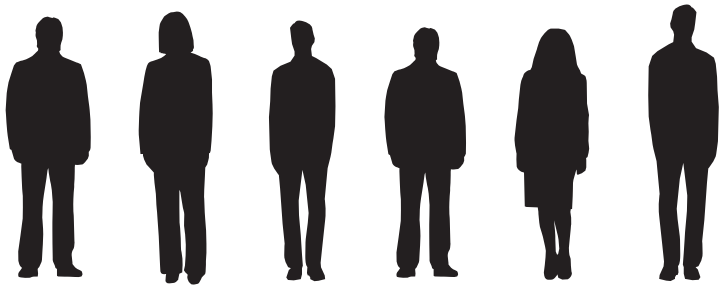
Part of doing this kind of trading is sitting in this kind of room...



Financial Trader

With a group of skilled people all sitting at the same big desk...

SHARED ENVIRONMENT



Fellow Traders + Shared Ways of Working

+

Dependable Enabling Technologies

And all of these technologies and applications available for immediate use...

Key *application envisioning* questions:

How could your team's insights into the realities and constraints of targeted knowledge workers' physical and cultural environments shape your application concepts? How might your computing tool meaningfully and valuably "fit" into these complex contexts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What size and variety of organizations might your team be targeting with your interactive application? How similar are these environments to each other?

How could specific cultural characteristics of targeted workers' environments, such as shared norms, values, and customs, impact the strategic direction of your team's computing tool?

How have these characteristics changed over time, and what directions are they trending in now?

What breakdowns in work practice are currently caused or aggravated by environmental factors? Could these breakdowns represent potential opportunities for your product?

How does the concentration or distribution of related physical spaces currently impact knowledge workers' practices?

How do physical contexts shape workers' communicative, cooperative, and collaborative efforts?

How are important work artifacts "located" within physical space and cultural zones? What understood norms surround their use in different environmental circumstances?

What attitudes do targeted knowledge workers have regarding their own mobility? What activities do they expect to be able to accomplish at various locations?

How might different models and understandings of these environmental factors allow your team to envision application concepts that could essentially "belong" in targeted contexts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A2. Workers' Interrelations and Relationships

Social interactions in knowledge work activities often involve multiple categories of organizational roles and outside stakeholders. The cultural characteristics of knowledge workers' social worlds can pose key challenges and opportunities for product teams as they attempt to outline appropriate and compelling design strategies.

Examples from three knowledge work domains:

> An architect typically works with other architects on her team, project managers and partners within her studio, a variety of specialized external consultants, and her clients. As a broad generalist, she has different goals, expectations, and methods of working with each of these groups, and she wants to use computing tools that will not get in the way of these differing approaches **(see illustration on next page)**.

> A financial trader typically works with other traders, back office support, several levels of management, and many business contacts outside his firm. The technologies and processes that his company has built up over time express underlying, top down — yet shared — norms and values about how these different groups should formally interact.

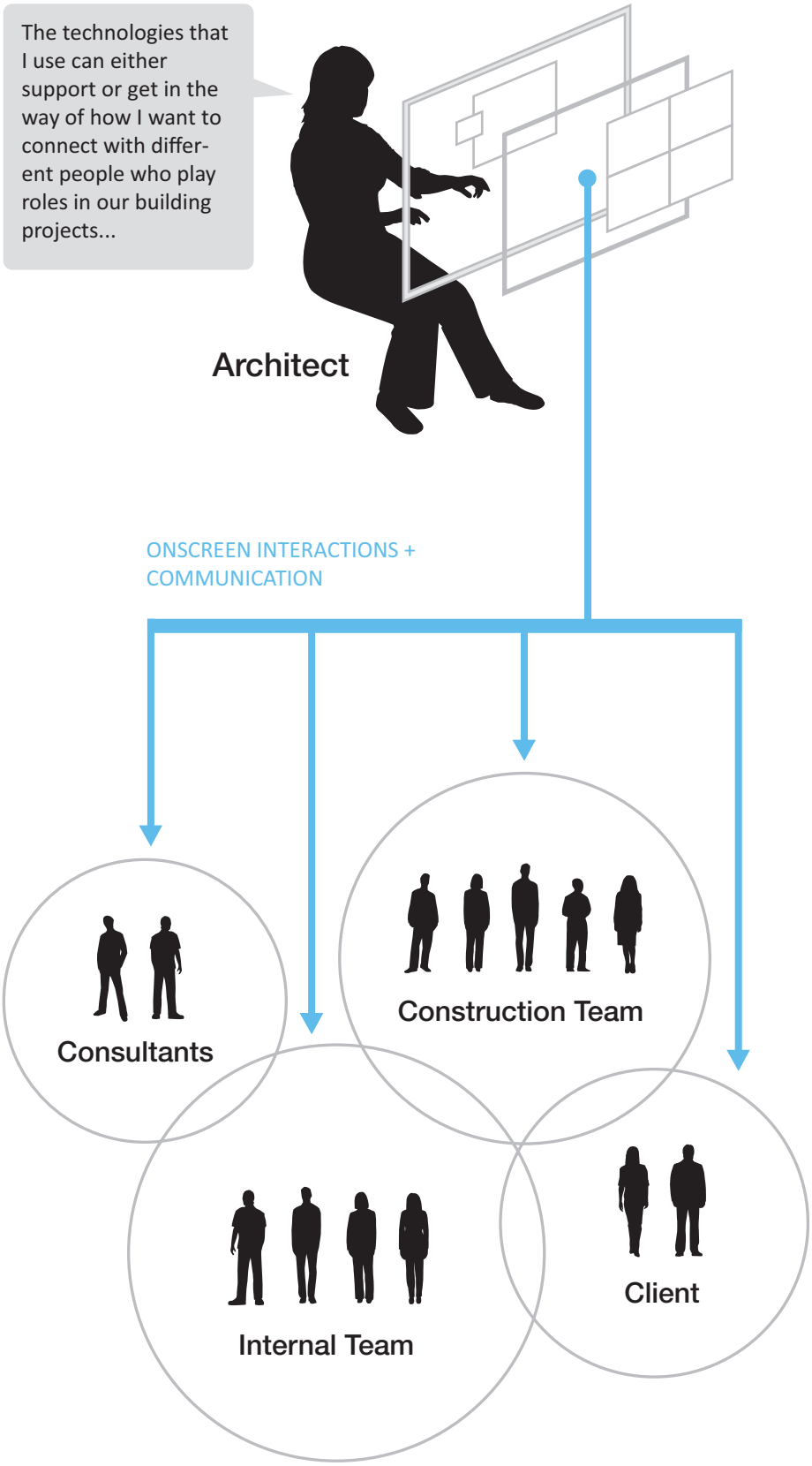
> A scientist typically works with other researchers in her clinical lab, the lab's technicians, representatives from regulatory bodies, a number of vendors, principle investigators at other labs, and members of the scientific community at large. As the head of her lab, she wants to have some measure of control over all of its key internal and public interactions.

Knowledge work is often performed within complex social spheres that contain a range of overlapping cultural expectations (A1). As part of everyday work practice, successful individuals can become skilled at acting within, and making use of, certain interpersonal relationships.

Product teams can model these relationships in search of valuable insights that could be meaningfully reflected in their divergent application concepts. Conventional professional practices, along with understood workflow and power structures within organizations, may dictate how different actors work together to accomplish certain outcomes (A4, C6). Additionally, local ways of working may arise organically from a shared grounding of implicit norms and customs, which can be reflected in divisions of labor (A7, A8) and resulting artifacts (B).

When product teams do not actively consider how the specifics of workers' social worlds might impact their emerging ideas about work mediation and application scope, opportunities to clearly situate a product in the context of these interpersonal networks can be lost. Applications that do not allow expected social interactions or reflect expected power relationships (A2, C5, B7) can be more difficult for workers to learn (D2, K2, K6) and may not be seen as useful or attractive options (D3, K3). These products may also not adequately support important cooperative or collaborative work practices (C7, G4) such as handoffs (G7, J3) and other forms of communication (J).

See also: A, B8, C8, M


The technologies that I use can either support or get in the way of how I want to connect with different people who play roles in our building projects...
Architect
ONSCREEN INTERACTIONS + COMMUNICATION
Consultants
Construction Team
Internal Team
Client

Key *application envisioning* questions:

How could your team's insights into the connectivities and qualities of targeted knowledge workers' relationships shape your application concepts? How might your computing tool usefully and meaningfully reflect these social realities?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How are workforces divided up in the organizations that your team might be targeting with your interactive application?

What roles do different groups of knowledge workers play in the context of different activities?

How do these groups of workers overlap and interrelate? How could your team characterize their goals and attributes based on observed relations in real world settings?

Which social network ties and interpersonal interactions are the most important for successful work practice?

Which ties do targeted workers enjoy and value?

Which interactions are problematic? Could these breakdowns represent opportunities for your product?

What directions are these interpersonal connections trending in? What changes in organizational relationships have occurred in the recent past?

What overriding management attitudes about workers' interrelations could influence the success of your computing tool?

How might different models and understandings of these social factors allow your team to envision application concepts that could improve valued interpersonal interactions for all involved?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A3. Work Practices Appropriate for Computer Mediation

Interactive applications can provide knowledge workers and their organizations more value in some activity scenarios than in others. To drive an appropriate and compelling application scope, product teams can balance the desire to usefully facilitate targeted workers' goals and practices with contemporary limitations of the computing medium.

Examples from three knowledge work domains:

> A scientist tests a variety of novel techniques to ensure that her laboratory is taking advantage of the latest clinical research methods. While she uses certain functionalities in her lab's onscreen applications to perform these tests, she does not expect these computing tools to support such open explorations to the same degree that they support high volume, standardized experiments **(see illustration on next page)**.

> A financial trader spends most of his day using interactive applications to accomplish predictable tasks. Since he knows that these tools can make important transactions somewhat impersonal, he often spends part of his day strengthening business relationships through informal phone and face to face chats.

> An architect begins her projects with free form sketching of potential shapes and ideas. She will not use her building modeling application, which emphasizes exacting details, to perform this very fluid early work.

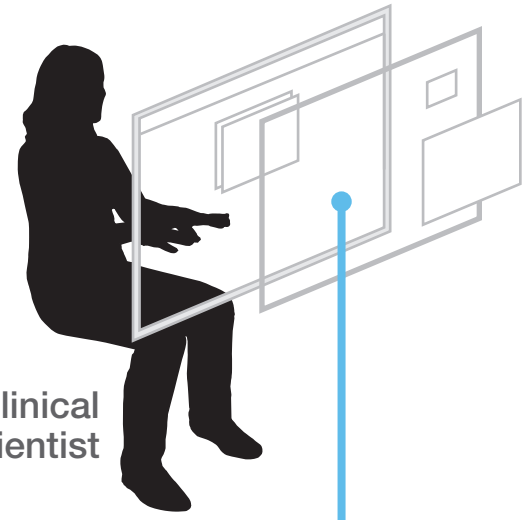
For a variety of reasons, not all knowledge work practices are well suited to being mediated by an interactive application. Workers may value their current, offline methods of accomplishing certain tasks or larger activities (A5) to an extent that they do and not want to change their proven customs. Even when people are open to certain changes, the limitations of contemporary computing may prove too constraining for some types of thinking work (D1). For example, conventional computing tools inherently standardize activities in ways that can restrict exploration (A4), and they typically support collaboration by offering highly individualistic actions within cooperative environments (C7, G4).

With knowledge workers' preferences and the limitations of current technologies in mind, product teams can carefully target activities where their application could desirably and feasibly provide value. Since workers may briefly use computing tools even in "intentionally offline" activities, teams can also respectfully envision more fleeting touchpoints (G5). These brief points of connection can sometimes serve as valuable opportunities to support smaller goals with tailored functionality, such as the ability to inform a decision by searching for related information (B8, I5).

When product teams do not actively consider whether targeted work practices are appropriate and conducive for onscreen interaction, resulting applications may contain extensive functionalities that are not particularly appreciated by knowledge workers. These products may be difficult to learn and "clumsy" in action (D2, K2, K6, K13). When organizations make such tools a standard part of their processes, workers may resent these technologies and limit their own use of them (D3, K12).

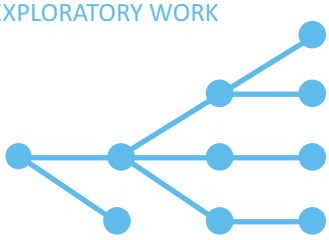
See also: A, C6, D4, E5, M1, M4

Not every part of our lab's scientific workflow should be supported by software designed specifically for it...



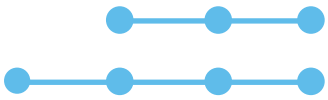
It's true that our lab's information management and analysis applications are always something we turn to when we are doing "production" work...

EXPLORATORY WORK



How does this method work? How might we use it in a study?

PRODUCTION WORK



How can we execute on this study plan? What findings are in its data?

Transition to use in a clinical study ▷

Activities understood as being too variable to be a functionality focus in primary software tools

Interactions that scientists expect to be a functionality focus in their primary software tools

But I don't expect those tools to support our leading edge, exploratory work.

When we are trying out new things, we often turn to more generalized tools, write our own rough code, or use scientific software in unintended ways...

Key *application envisioning* questions:

Where in your team's big picture characterizations of knowledge workers' activities do you see potential value and possibility for useful and meaningful mediation by a computing tool? From a vantage point that emphasizes targeted workers' mental efforts, where is there less potential value and possibility?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What portions of their work practices do targeted individuals and organizations not want to move onscreen? What portions would they like to have supported by an interactive application? Why?

How might contemporary computing be too closed, individualistic, and constraining for the knowledge work that your team is targeting?

Which work practices do not inherently lend themselves to being mediated by a near term computing tool?

Which work practices could be ripe for on-screen support, facilitation, and enhancement?

What larger trends and advanced analogies in technology adoption could valuably inform your team's decision making about which activities to target?

Are there any opportunities for your application concepts to support small portions of otherwise "off screen" work, rather than larger expanses of work practice?

How might your team model and use these understandings to envision functionality concepts, potential application scopes, and larger strategic directions for your product?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A4. Standardization of Work Practice through Mediation

When interactive applications introduce new possibilities in support of knowledge work practices, they often also introduce new levels of standardization. Product team can envision appropriate levels of freedom and constraint in their application concepts, which can range from a slight narrowing of available choices to the restrictive organization of entire activities.

Examples from three knowledge work domains:

> A financial trader used to communicate about certain topics through a variety of different channels, but now he frequently uses his new trading application instead of reaching for other options. It has functionality that allows him to quickly send targeted messages to relevant parties, and he likes the idea of his group standardizing their approach to communication **(see illustration on next page)**.

> An architect used to have different approaches to adding construction notes to different types and scales of drawings. When her studio made the switch to using a building modeling application, which has very different implications and opportunities for these notes, she worked to inform and educate external colleagues about a new set of notation standards.

> A scientist sets up procedures for her lab technicians to follow. While these procedures have always been consistent, the introduction of her lab's new information management application has facilitated new levels of useful standardization that had previously been too difficult to achieve.

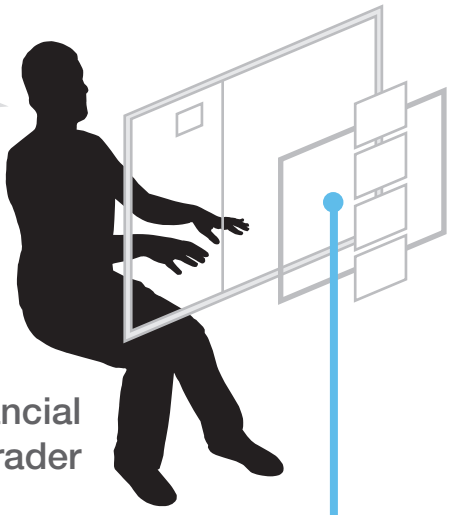
Interactive applications inherently contain some standardizing constraints. For example, data attributes may have a predefined list of valid options, and navigation pathways between functional areas may be strung together in meaningfully predetermined ways (C4). Some designs for computing tools are more directive than others, and channeling constraints can have different levels of mutability, ranging from somewhat flexible to highly fixed (K6).

Product teams can sketch standardizing constraints that are useful and well suited to targeted tasks and larger activities. Depending on standardization goals, a routine knowledge work procedure could be supported with a set of random access tools in an open application workspace (A6, G2), an entirely fixed interactive workflow (C6, D4), or even an automated procedure (E3, E4). When incoming requests for standardization are inconsistent (A2, A7, A8), teams can map consistencies and variabilities in order to envision default approaches, along with methods of customizing those defaults to meet local practices and individual needs (C8, D1). However, sometimes effective standards simply cannot be defined.

When product teams do not actively consider how implicit or explicit standardization might impact their emerging ideas about work mediation and application scope, opportunities to provide valuable inflexibilities can be lost. When applications contain inappropriate standardization, they can create frustrating and unpersuasive limitations on action, potentially leading to difficulties in adoption (K) and excessively effortful workarounds (D2, D3).

See also: A, B5, E, F, G1, J6, L2, M1, M4

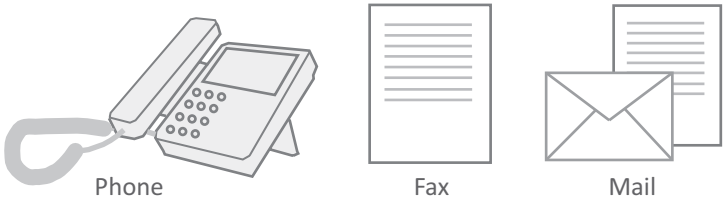
Communication is what trading is about, and our group tries to keep our interactions with the outside world as consistent as possible...



For example, we use an automatic form to rapidly email clear and legible trade cancellations...

Which is very different from how we used to tell our trading partners about cancelled deals...

CHANNELS USED PRIOR TO STANDARDIZATION OF WORK PRACTICE



Everyone in our group did it differently, which was confusing and eventually drove us to create a useful standard...

Key *application envisioning* questions:

Where in your team's big picture characterizations of knowledge workers' activities could inherent standardization be valuable in a supporting computing tool? Where might targeted individuals and organizations view standardization as restrictive and problematic?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which standardizations of work practice do targeted individuals and organizations currently value? Why?

Where have they intentionally avoided standardization? Where do they disagree on the topic?

What value does standardization provide in current practices?

Who defined current standardizations? How were they introduced?

Which areas of work practice are trending toward more standardization? Which are trending toward less?

How are agreed upon work practices formalized into structured work processes within targeted organizations? What might your team learn from these transitions?

Where could conflicting standardization requests make it difficult to define useful onscreen support? At what point are requests too diverse for a single computing tool to be effective for a majority of users?

What advanced analogies about standardizations in other fields could valuably inform your team's strategic ideation?

How might your sketched functionality concepts maintain or expand upon existing, useful standardizations?

What operations, tasks, or even entire activities that your team is considering for your product's scope will likely require further standardization in order to be supported effectively?

Which parts of your sketched application concepts could imply further standardization by design? Could these constraints be a hindrance or will they meaningfully direct interaction and work outcomes?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A5. Interrelations of Operation, Task, and Activity Scenarios

Knowledge workers' granular actions can be categorized as operations, which overlap and interrelate into larger tasks, which themselves overlap and interrelate into the larger unit of activities. Explicit models of these multi-tiered relationships can help product teams envision interactive applications that are much more than haphazard collections of unconnected, discrete functions.

Examples from three knowledge work domains:

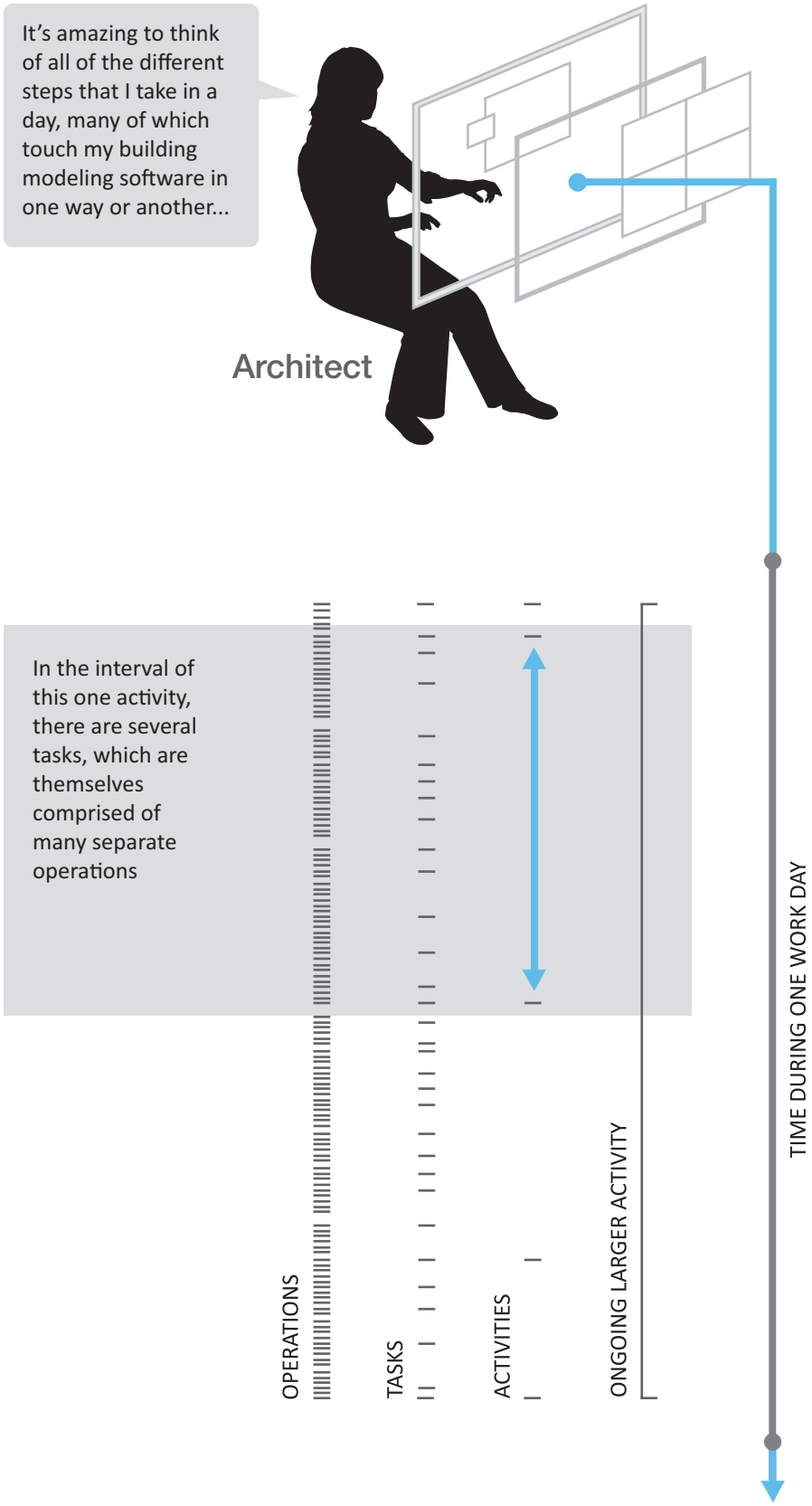
> An architect performs many small operations in her building modeling application, progressively completing separate tasks that incrementally advance the project. These individual advancements, in conjunction with her colleagues' contributions to the same model, result in a series of iterations, which eventually result in a complete and approved design **(see illustration on next page)**.
>
> A financial trader performs a number of steps while completing every trade. These individual trades contribute to his larger goal of advancing the profitability of his firm by maximizing the value of his own transactions.
>
> A scientist analyzes the clinical data generated by her lab technicians after each round of their experiments. These individual analyses accumulate into a study's findings, which then lead to further studies, in a chain of research that contributes to the accumulated knowledge of her clinical field.

In the process of rationalizing knowledge work for system design, product teams inevitably break down larger work practices into smaller pieces. They may characterize segments of work by inputs and outputs, the actors involved (A2), related goals, and many other factors (J3). While this deconstructive approach can be a key method for developing meaningful understandings of workers' behaviors, it runs the risk of severing inherent linkages that can be essential for effective envisioning of useful and usable computing tools (C4, G1).

Product teams can connect characterized units into networks and tiered hierarchies that reflect workers' current and desired practices. They can recognize that when they envision a specific activity as part of their application's scope (A3, A9), they are going to have to support at least some of its related tasks and operations. Teams can discover that these linkages between units may not be exclusive, so that, for example, the same task can be tied to two different activities, with slight variations based on differences in context (A7, A8). They may also see that the interrelations inherent in work practices could suggest, for example, a basis for automated functionality (E3, E4) or connectivity with other technologies (B8, K8, K9, K10).

When product teams do not actively consider how the interrelated nature of workers' practices might impact their emerging ideas about work mediation and application scope, opportunities to envision clearly defined, easily navigable, and functionally appropriate products can be lost (C1, C2). Considering these interrelations can be particularly important when teams are creating novel tools that do not have core, established conventions to fall back on (F2, L2).

See also: A, B4, B5, C6, F1, G5, K, M1, M4


It's amazing to think of all of the different steps that I take in a day, many of which touch my building modeling software in one way or another...
Architect
In the interval of this one activity, there are several tasks, which are themselves comprised of many separate operations
OPERATIONS
TASKS
ACTIVITIES
ONGOING LARGER ACTIVITY
TIME DURING ONE WORK DAY

Key *application envisioning* questions:

From a vantage point that emphasizes knowledge workers' mental efforts, how might your team break down your big picture characterizations of targeted workers' practices into a useful and meaningful hierarchy of activity, task, and low level operation elements?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which operations are so discrete that they probably do not need to be included in your envisioning process? How much detail is too much detail when thinking about a foundational model that your team can use to sketch potential design strategies and application concepts?

What user goals and other attributes might your team capture for each operation, task, and larger activity in your emerging rationalizations of knowledge work?

How should the discrete, individual elements within your team's models of current and desired work practices overlap, nest, and interrelate?

Could individual operations map to more than one task, or are they strictly hierarchical?

Could individual tasks map to several different activities?

Could individual activities map to other, larger activities?

How might the mapping of an individual work element to multiple situations change how it is practiced under different circumstances? Where could variations based on these mappings be drastic enough to call them out as different practices?

How might different scenario flows through your team's rationalized maps of work practice drive different requirements for functionality concepts?

Which threads and mappings in your models could be essential for envisioning your application's conceptual model, interaction model, and pathways for goal directed wayfinding?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A6. Open and Emergent Work Scenarios

Some knowledge work tasks and larger activities involve solving complex, undefined problems where workers' goals and methods evolve within unfolding pathways of effort. These emergent scenarios can be supported by interactive applications that present useful flexibilities, which product teams can envision as largely unsequenced but interrelated patterns of mediated work.

Examples from three knowledge work domains:

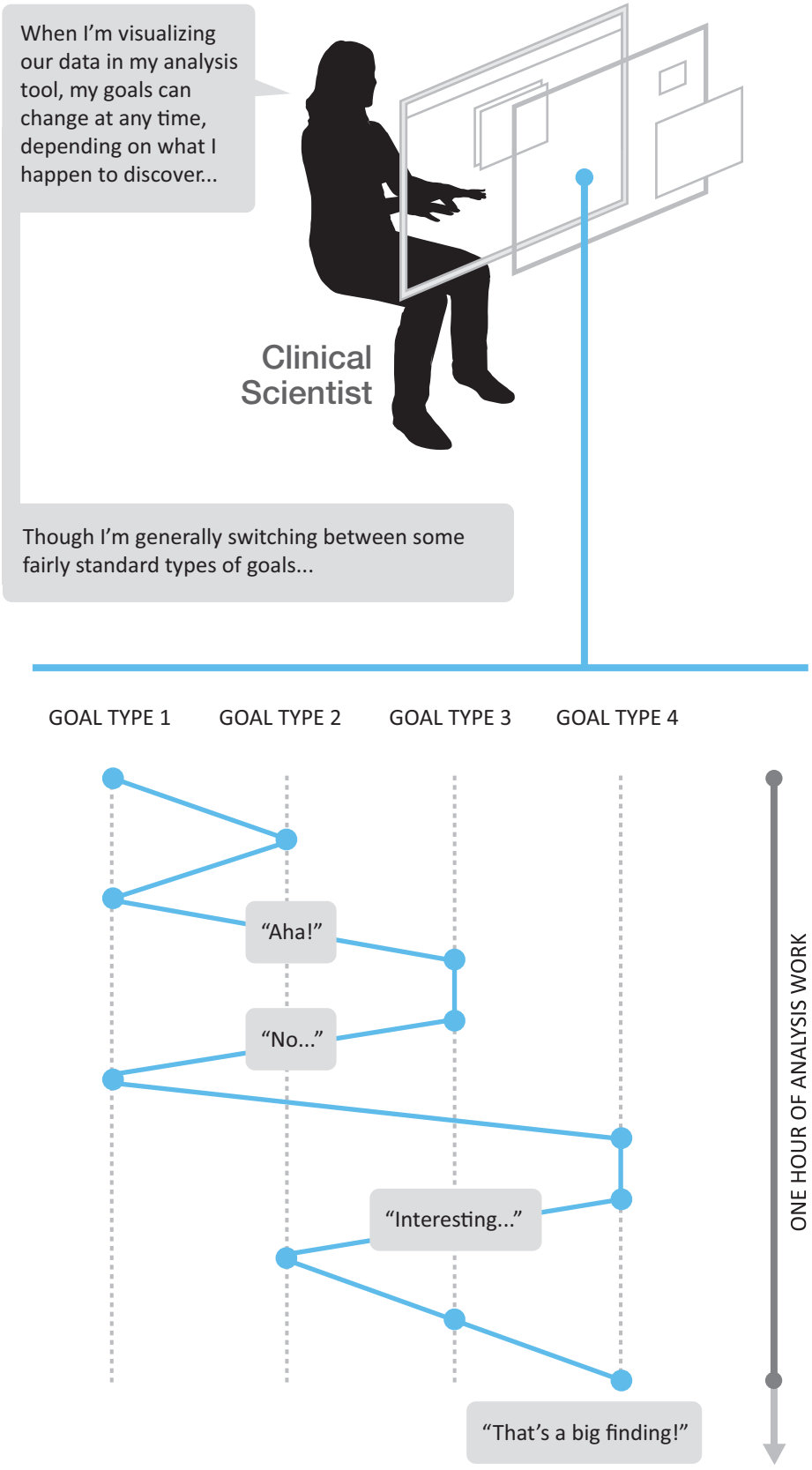
> A scientist's use of her analysis application is highly contingent on what trends she discovers in her lab's clinical results. Within the tool's data visualization functionalities, her goals can change drastically based on the patterns that appear after each visual transformation that she explores **(see illustration on next page)**.
>
> An architect is working in her building modeling application on a floor plan for a hospital's critical care ward. She tries out a number of different rough layouts that could meet the project's requirements, evolving her own criteria for a successful solution as she explores different ideas.
>
> A financial trader's work is primarily composed of frequent, brief, discrete, and habitual actions. However, some parts of his work are often not so routine, such as conversations about problematic trades or large potential deals, both of which can follow irregular processes and require unpredictable amounts of time.

Some types of knowledge work are practiced without step by step procedures or even high level road maps. Workers may begin these practices with clear goals in mind, but their intentions can evolve as outcomes unfold through a progression of actions. To successfully accomplish these scenarios, individuals can become highly skilled at recognizing patterns, situationally turning to supplemental resources and tools (G5, K8, K9), making meaning, testing hypotheses (F8, F9, I2, I3), revising their expectations and understandings, and defining success (L1).

The variations that stem from open and emergent ways of working can be difficult for product teams to appropriately capture in their shared, rationalized models (A7, A8). In some cases, a single model of these work practices can cover a critical mass of important variations. In many other cases, teams may benefit from creating models that represent a "cloud" of potential scenarios — an interrelated network of largely unsequenced patterns of action.

When product teams do not actively consider how open and emergent scenarios might impact their developing ideas about work mediation and application scope, resulting products may lack necessary flexibilities (A9). In the name of standardization (A4), product teams may crystallize processes based on inadequate understandings of complex realities (B8, C8), resulting in applications that can be difficult for workers to adopt and use (D2, D3, G1, K). At their worst, these hindrances to open and emergent work can be evidenced in the overall framework of a computing tool (C1, C2), which can be an excessively difficult issue to correct in implemented products.

See also: A, B4, G6, H, I5, K3, K6, K11, M1, M4


When I'm visualizing our data in my analysis tool, my goals can change at any time, depending on what I happen to discover...
Clinical Scientist
Though I'm generally switching between some fairly standard types of goals...
GOAL TYPE 1
GOAL TYPE 2
GOAL TYPE 3
GOAL TYPE 4
"Aha!"
"No..."
"Interesting..."
"That's a big finding!"
ONE HOUR OF ANALYSIS WORK

Key *application envisioning* questions:

What areas of your team's emerging models of work practice are accomplished through open and emergent pathways of knowledge work rather than strict, process oriented action? From a vantage point that emphasizes targeted workers' mental efforts, how much functional flexibility could be required to valuably support these cases?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What tasks or larger activities, within the scope of work that your team is investigating, take shape through the improvisational structure of workers' practices?

What do targeted workers accomplish in these open and emergent scenarios and variations?

What are the initiating goals in each of these cases? How can those goals evolve through different series of actions?

Is the knowledge work domain that your team is targeting trending toward more improvisation or toward further specialization of defined processes and roles?

How do targeted individuals and their organizations view the importance of open and emergent practices? Do they wish they were more standardized? Do they value their openness?

What situations in these improvisational scenarios trigger workers to make decisions about subsequent approaches and actions?

What are the most important points of flexibility for your team to consider when attempting to support these work practices?

What other patterns and regularities can your team find in these "clouds" of potential scenarios? How might you use these insights to ideate useful and meaningful functionality concepts?

How could support for these practices impact the overall scope and frameworks of your application concepts?

How far might your team push certain flexibilities for open and emergent practices before the interaction clarity of your sketched computing tools begins to break down?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A7. Collaboration Scenarios and Variations

Even apparently individualistic knowledge work practices can have key collaborative, or at least cooperative, scenarios and variations. By actively envisioning how these cases might be supported by an interactive application, product teams can avoid common and disruptive pitfalls in their approaches to mediating work.

Examples from three knowledge work domains:

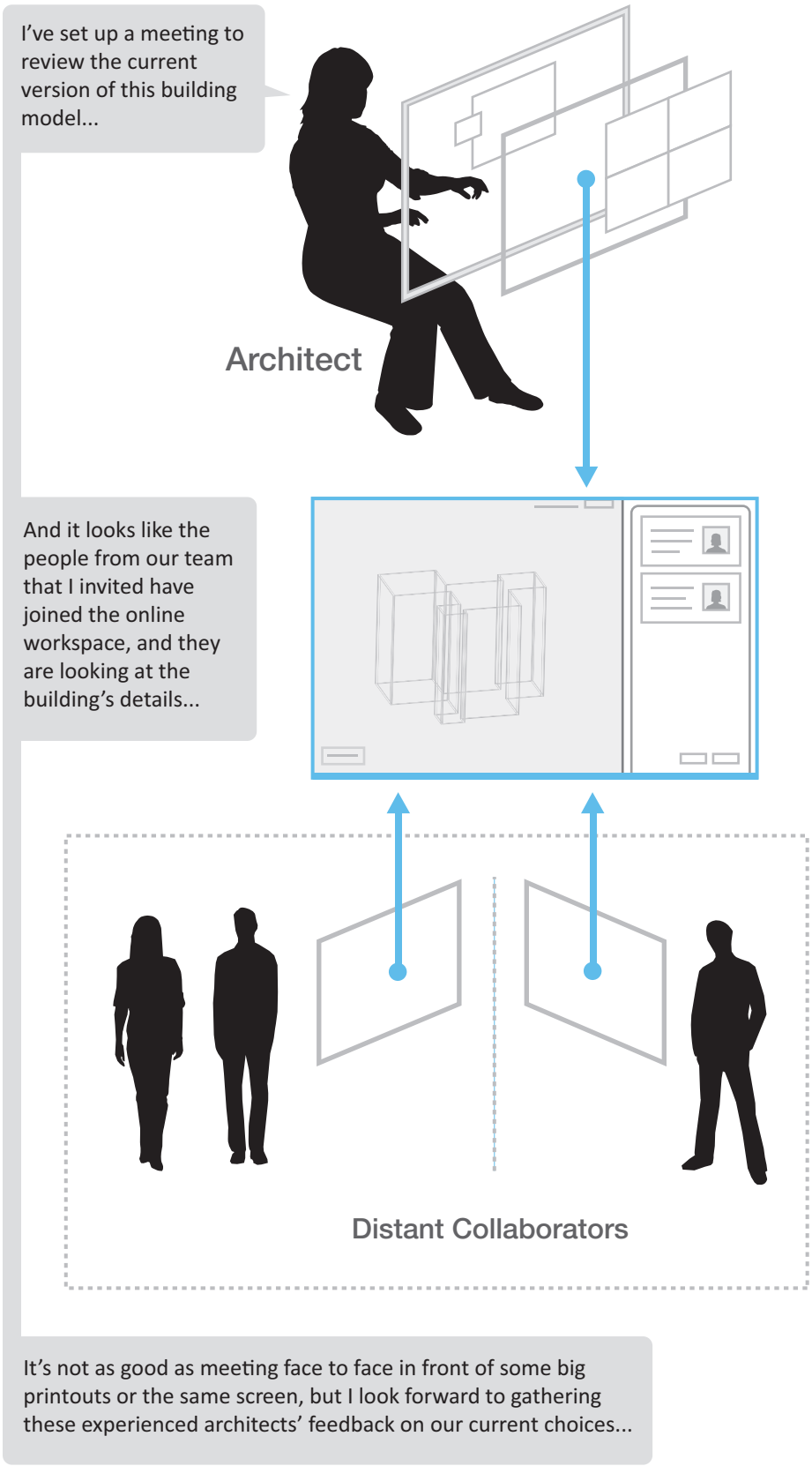
> An architect frequently reviews her project work with one or more colleagues in her firm, either formally or informally. While this used to typically occur face to face, writing on paper printouts, her company's new building modeling application now allows her to meet online with team members from different global offices in a shared, highly visual workspace **(see illustration on next page)**.
>
> A financial trader sometimes shares the details of important pending deals with other traders in his group. Their firm's trading application allows him to save draft proposals of large, complex deals to a shared location where his colleagues can access and work on them.
>
> A scientist sets up her clinical research lab's information management application in a way that allows certain lab technicians to "own" certain tasks. She makes an exception for quality checking procedures, which will require the input of two separate lab techs.

Collaboration in knowledge work can range from asking quick questions to spending long hours actively working with colleagues, either in person or at a distance (A1). People may recognize some tasks or larger activities as explicitly collaborative, whether that collaboration takes place in real time or asynchronously (F1, J2). Even in areas of work practice where individuals do not feel that they are directly collaborating, they are often cooperatively completing their own parts of a larger process while sharing certain elements of their organizational contexts (C5, G7, J3, J4).

Variations that stem from collaborative ways of working can be difficult for product teams to meaningfully rationalize (A4, A6, A7, A8). In some cases, a single model of how a product could mediate knowledge work can cover a critical mass of important variants. In many other cases, teams may benefit from creating multiple models of the same area of work practice in order to usefully and appropriately describe specific instances of collaborative, or at least cooperative, action.

When product teams do not actively consider how the collaborative aspects of knowledge work might impact their emerging ideas about work mediation and application scope, resulting products may not be adopted by individuals and organizations that place a high value on shared, convivial work (K). Applications' frameworks (C1, C2) may mistakenly emphasize individualistic directives over cooperative interactions, inhibiting both the distribution of effort and meaningful visibility into others' actions (C7, G4). Such frameworks can also contribute to the likelihood of human error (C9, G3) and drive workers to perform excessively effortful work arounds (D2, D3, D4).

See also: A, B5, B6, B7, B8, H2, H3, J1, J5, M1


I've set up a meeting to review the current version of this building model...
Architect
And it looks like the people from our team that I invited have joined the online workspace, and they are looking at the building's details...
Distant Collaborators
It's not as good as meeting face to face in front of some big printouts or the same screen, but I look forward to gathering these experienced architects' feedback on our current choices...

Key *application envisioning* questions:

What areas in your team's emerging models of knowledge work practice can involve collaborative, or at least cooperative, action? How might attempting to mediate these complex practices impact the functional forms and overarching strategic directions of your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What tasks or larger activities, within the scope of work practice that your team is investigating, are inherently collaborative?

What parts of knowledge work that could otherwise be considered individualistic have collaborative or cooperative variations?

What do targeted workers accomplish in these scenarios and variations?

What are their goals in each of these cases?

What breakdowns in work practice are currently caused or aggravated by cooperative and collaborative interactions? Could these problems represent potential opportunities for your team's product?

Is the knowledge work domain that you are targeting trending toward more collaboration or toward further specialization of defined processes and roles?

How do targeted individuals and their organizations view the importance of collaborative practices? Do they wish they were more individualistic? More collaborative?

What specific aspects and effects of collaboration do workers perceive as valuable? Which are inherently important for successful outcomes?

What other patterns and regularities might your team find in shared, convivial practices?

How might you use these insights to ideate useful and meaningful functionality concepts?

How could support for these practices impact the overall scope and frameworks of your application concepts?

How far might your team push certain flexibilities for collaborative scenarios and variations before the interaction clarity of your sketched computing tools begins to break down?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A8. Local Practices and Scenario Variations

Knowledge workers may continually refine their approaches to certain tasks and larger activities in order to meet their local needs, performing adaptive variations based on recognized contingencies. Product teams can envision how diverse yet essential variations in workers' practices might be supported by thoughtful flexibilities in their application concepts.

Examples from three knowledge work domains:

> A financial trader has worked at three different firms in the last five years, using the same trading application in each organization. Although each firm had slightly different ways of accomplishing the same goals, the trading tool consistently displayed the right kinds of flexibility to be effective in each environment **(see illustration on next page)**.
>
> A scientist's use of her analysis applications depends on the purpose and methods of the particular clinical studies that her lab is currently conducting. However, looking across the different types of studies that her lab has recently pursued, she thinks that she typically performs different "flavors" of the same essential analyses.
>
> An architect meets with her team at the end of every project to discuss potential process improvements. Looking back across two years, she sees that her studio's detailed approaches to working have evolved more than she had realized.

Examining knowledge work across a number of organizations, there can be can be major variations in how different individuals and groups accomplish the same types of work practice (A1). Even organizations operating in highly similar fields can have very different goals, established processes, observed methods, and barriers to success. Within a given workplace, people may have developed several different ways to accomplish certain goals based on recognizable cases.

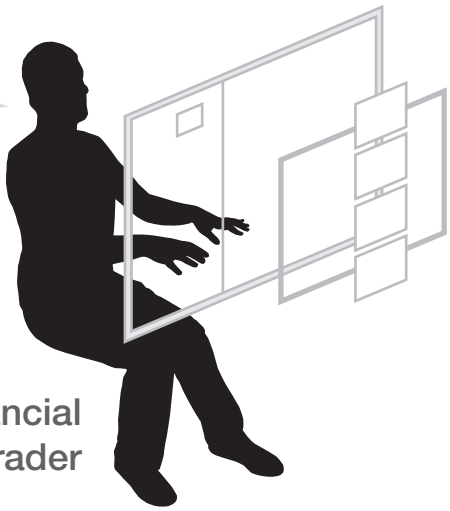
Branches that stem from local and variable approaches to work can be difficult for product teams to meaningfully distill into shared, rationalized models (A4, A7). In some cases, a single model of how a product could mediate knowledge work can cover a critical mass of important variations. In many other cases, teams may benefit from creating multiple models in order to usefully and appropriately describe important categories and families of related scenarios.

When product teams do not actively consider how local practices and scenario variations might impact their emerging ideas about work mediation and application scope, resulting products may lack needed flexibilities for some locales. When presented with applications that do not adequately reflect their current practices (K3), knowledge workers may not want to change their well known ways of working in order to make use of new tools (D2, D3, K). Even when a product's implied changes are desirable, some established, "home grown" approaches may be exceedingly difficult to update.

Conversely, too much emphasis on supporting diversity in work practices may lead to unnecessary flexibility that can reduce learnability (K2, K6) and interaction clarity (G1) for more critical, common, and frequent scenarios (A9).

See also: A, B, C8, E, F1, F2, I1, M

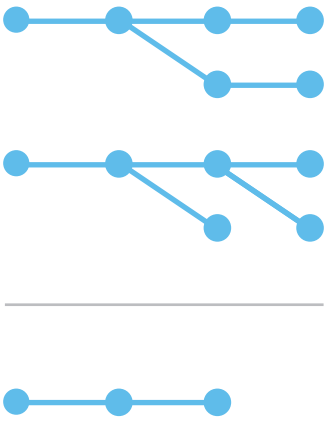
This trading tool is remarkably adaptable. I think that the people that designed it really know the small but important differences in how people trade...



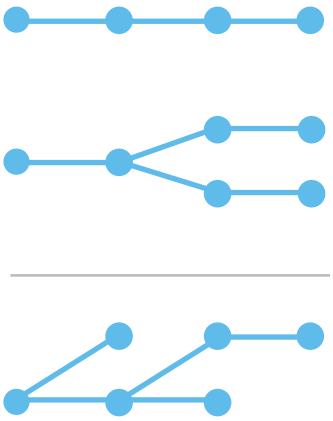
Financial Trader

All of the firms that I've worked at have been able to successfully work with the same software in their own slightly different ways...

PREVIOUS FIRM



CURRENT FIRM



VS

At my last job, there was a general emphasis on allowing us traders to do things our own way, which gave us just enough rope to hang ourselves...

But when it came to negotiating, they had specific processes that they wanted us to follow...

At my current firm, they have thought a lot about where standard processes could be valuable and provided good tools to help us get to those standards...

But in negotiation, they give us a lot of freedom...

Key *application envisioning* questions:

How might your team's emerging models of knowledge work practice call out key local variabilities between and within targeted organizations? Where in your mapped understandings could different scenarios for accomplishing the same goal be important? How might those differences impact the overarching functional forms and strategic directions of your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What tasks or larger activities, within the scope of work practice that your team is investigating, are performed differently in different locales and situations?

What variabilities stem mainly from local differences in practices, as seen when looking across targeted organizations?

How do common differences in workers' personal behaviors and preferences create categorical variations in work practice? What circumstantial cases can drive important differences in workers' approaches to accomplishing a goal? What can cause these divergent branches from a "normal" practice?

How do targeted individuals and organizations view the importance of their own ways of accomplishing work? Are they aware of other ways of doing things?

Are the variabilities that your team has identified trending toward more consolidation or further division?

Which variations could be thought of as critical or frequent enough to model as separate but related work practices? Which local practices are uncommon? Which are frequent or seen as critical by targeted workers?

Which variations do people value just as much as the "normal" flows of their own work practices? Nearly as much as?

What practices might individuals and organizations be open to changing in order to make use of a valuable new product? What offerings could provide that level of value?

What other patterns and priorities could your team identify in these variations on workers' practices? How might you use these insights to ideate useful and meaningful functionality concepts?

Are local practices and scenarios variations so heterogeneous and diverse as to make a single application solution difficult to envision?

How could support for these practices impact the overall scope and framework of your team's application concepts?

How far might your team push flexibility for local practices before the interaction clarity of your sketched computing tools begins to break down?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

A9. High Value Ratio for Targeted Work Practices

Not all of a product team's sketched functionality concepts have the same potential to provide compelling utility in knowledge work. To promote usefulness and cohesive design strategies in their application concepts, teams can parsimoniously target certain work practices by including related, high value functionalities and downplaying or eliminating unrelated, lower priority options.

Examples from three knowledge work domains:

> A scientist had previously analyzed her lab's clinical data by using small portions of several different applications. Her new analysis application contains all of those useful functionalities in a single product, while "cutting the fat" of options that researchers like her never use (see illustration on next page).
>
> A financial trader wants developers of a new trading application to focus on the core tasks that he repeats throughout his work day. While there are a lot of other features he would "like" to have, he does not want any of them added to the new tool if their inclusion would take away from exciting and appropriate support for the core of his trading work.
>
> An architect uses different functionality in her building modeling application at different intervals of a building project's life span. While she feels that she has used a majority of the tool's available options at one point or another, during any one interval of a project she uses only a concentrated subset of its features.

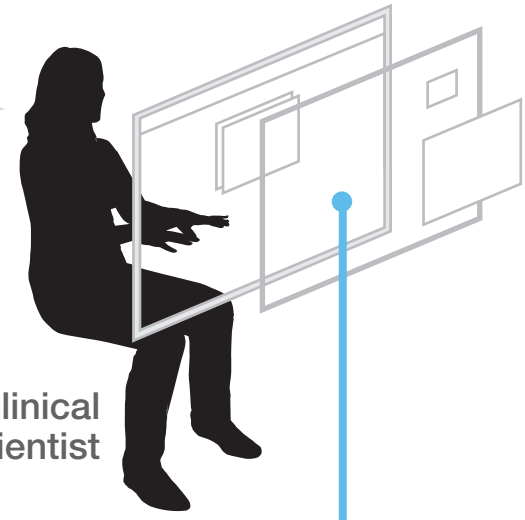
Every interactive application has a limited scope and is intended for use in a certain range of circumstances (A). Similarly, each application concept that a product team envisions reflects a set of design priorities that can be compared with and situated within larger spaces of possibility. Inevitably, functionality concepts that support a subset of tasks and larger activities become more substantially developed, while other concepts wither or disappear (A3, A5).

To arrive at an appropriate functional scope and refined design strategy, product teams must have a clear understanding of the goals, pain points, unmet needs, and measures of success that are prevalent in their targeted markets. In knowledge work domains with extensive, highly enmeshed, and frequently practiced groupings of tasks, appropriate application concepts may become relatively large and complex (C4). Conversely, appropriate concepts for narrowly targeted, infrequent roles in work practice can often benefit from a reductive simplicity (E3, E4) that promotes directive learnability and interaction efficiency (A4, K2, K6).

When product teams do not actively consider how a computing tool's potential options could provide differential value in mediated work practice, resulting products may suffer from an overabundance of features, a condition that Donald Norman has termed "featuritis." This overabundance may be caused by teams directly translating workers' requests into functional requirements. A lack of clear priorities can also lead teams to under develop critical functionality, potentially resulting in products that are seen as unattractive (K3) and difficult for workers to adopt and use (D2, D3, G1, K).

See also: B1, C1, C2, K10, L, M1, M4

So many scientific applications are huge and generic, filled with bloat, or small and focused but missing so much of what our lab needs for our own research goals...



Clinical Scientist

PREVIOUS USE OF SEVERAL ANALYSIS APPLICATIONS

Most of the functionality in each of several applications was left unused by the laboratory team.

~10% ~5%

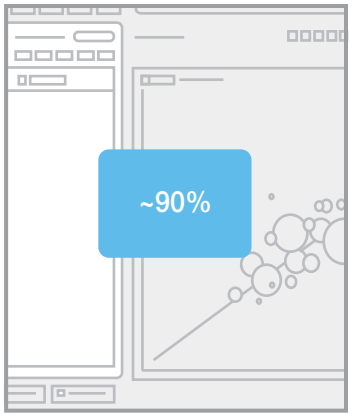
~15% ~20%

~5% ~10%

VS

CURRENT USE OF SINGLE ANALYSIS APPLICATION

Main analysis application contains few options that the laboratory team does not use.



For example, it used to be that we would have to simultaneously use bits and pieces from different analysis applications in order to accomplish what we wanted...

And then we found our new analysis tool, which is designed for our type of research, and meets I'd say 90 percent of our needs...

Key *application envisioning* questions:

Which areas of knowledge work practice might your team want to target with your product? From a vantage point that emphasizes workers' mental efforts, which selective assembly from among your sketched functionality concepts could provide compelling value in targeted work, while at the same time coalescing into a sensible application concept that embodies a well resolved design strategy?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which specific operations, tasks, and larger activities will your team target with your computing tool?

Which elements in your mapped understandings of work practice will you intentionally exclude from your application concepts?

Which of your sketched functionality concepts emerge as the essential, valuable, and desirable "core" that could support these targeted practices? What design strategies could that aggregation imply?

Which of your team's functionality concepts could be prioritized as secondary? As tertiary? As potentially unnecessary?

Which of your envisioned directions for your computing tool map to one or more established product genres in your targeted markets?

If your envisioned product is not representative of a known genre, will workers perceive its key offerings as interrelated and cohesive given the context of their own practices?

What analogies might your team draw from established product genres in other, seemingly unrelated markets?

What are the overarching stories of your team's emerging application concepts?

What could these narratives mean for your product's evolving brand and positioning in the market?

What functionalities do competing products provide that workers may expect from your team's interactive application?

What larger product marketing, technology, and design trends could influence your team's ideas about application scope?

Where could reductions in functional scope drive desirable simplicity in your application concepts?

How might ideas about product scope inform your team's envisioning of an appropriate application framework, learnability requirements, and other key design considerations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B.

Defining Interaction Objects

Valued computing tools can present clearly articulated and understandable collections of onscreen objects that knowledge workers can act upon, with, and through.

Designing such clarity requires deliberate mapping and careful simplification.

During *application envisioning*, product teams can sketch and explore the interaction objects that users might encounter in different scenarios of mediated work.

By taking time to generate diverse ideas about users' potential experiences of onscreen entities, teams can codify essential characteristics, behaviors, and relationships.

Within a product team's emerging concepts for mediating knowledge work, there are both actions and implied or explicit recipients of those actions. In some cases, the recipient of an action may be an onscreen tool that workers can act either on or through. When product teams do not thoughtfully frame and flesh out these primary onscreen objects, resulting applications may present workers with inconsistent, unfamiliar, and confusing data structures that feel as if they must be learned "from the ground up."

Legible interaction objects can leverage workers' existing expertise by directly referencing specific artifacts that are currently found in their work practices. By drawing meaningful connections to known constructs and material culture, applications can trigger useful expectations in workers that may help them to understand what can be done to and with corresponding onscreen items.

There are a number of specific issues that may arise when work practice transitions from dealing with material artifacts to dealing with intangible interaction objects. Many of these issues can be the result of reducing or eliminating important cues that workers normally read from artifacts' physical placements and visible forms. To actively address these potential problems, product teams can design key cues back into onscreen objects based on careful consideration of usage scenarios.

This category contains 10 of the 100 *application envisioning* ideas in this book:

B1. Named objects and information structures

B2. Flexible identification of object instances

B3. Coupling of application and real world objects

B4. Object associations and user defined objects

B5. Object states and activity flow visibility

B6. Flagged variability within or between objects

B7. Object ownership and availability rules

B8. Explicit mapping of objects to work mediation

B9. Common management actions for objects

B10. Object templates

Product teams can use these ideas to explore knowledge worker's potential experiences of the interaction objects in their application concepts. Given the inherent abstraction of computing environments and the limited space of workers' screens, early ideation on this topic can promote the development of conceptually clear, consistent, and actionable focal points within computing tools.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Establishing an application framework" (C), "Enhancing information representation" (F), and "Working with volumes of information" (I) categories.

B1. Named Objects and Information Structures

Knowledge work applications can support specific work practices with named interaction objects that are equivalents of familiar workplace artifacts. In addition to incorporating existing domain ideas and entities, product teams may need to introduce new objects into workers' vocabularies and practices in order to meaningfully enable certain functionality concepts.

Examples from three knowledge work domains:

> A scientist sets up a new clinical research study in her lab's information management application. She creates a study file, a revised lab automation procedure, and onscreen instantiations for several clinical samples and test tubes that are physically present in her lab **(see illustration on next page)**.
>
> A financial trader's work primarily focuses on individual trades, though his trading application subdivides each deal into several different subcomponents that are meaningful for certain tasks.
>
> An architect uses various modeling tools, standard 3D shapes, templated components, and many other onscreen elements to design buildings with her building modeling application.

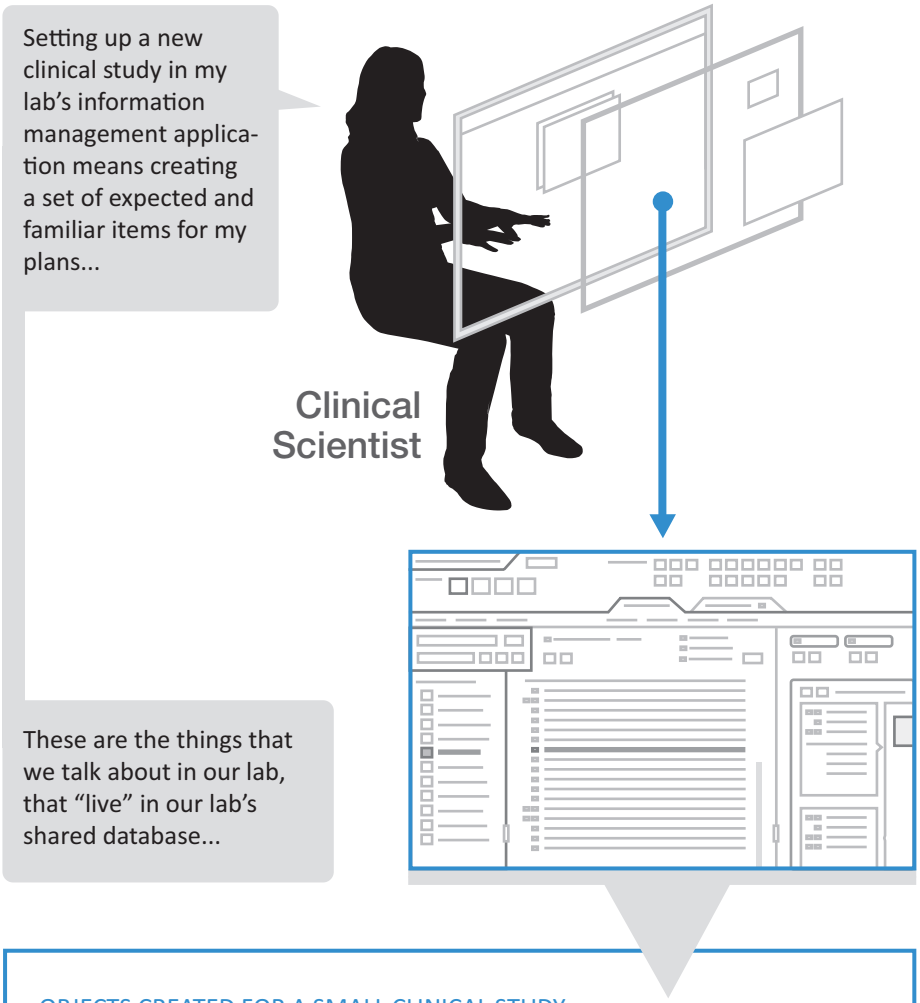
When knowledge workers act "through the screen" of an interactive application, they are typically acting on specific, named objects that are framed by and made visible through the product's display. These named, visible "pieces" of an application can be central to its underlying conceptual models (C1) and can activate workers' deep seated understandings and skills.

Product teams can adapt many interaction objects from existing tools, resources, work products (L1), and other artifacts that have historical trajectories of use within a knowledge work domain (A). In order for these conventional objects to make sense in a computing context, they may require substantial transformation and thoughtful reframing (K5). For example, a single artifact may need to broken into multiple interaction objects in order to support certain actions (B4, G2). To maintain recognizability, adapted objects that undergo considerable redesign can reference conventional visual forms (F2) and useful iconic resemblances (L3).

Existing domain objects may not adequately support some of a product team's concepts for mediating work. Teams must commonly envision new interaction objects to represent useful system concepts that have no previous corollary in offline work, such as customization settings (C8) or object templates (B10).

When product teams do not actively consider the menageries of interactive objects that form the primary "materials" of their sketched application concepts, opportunities to drive learnability and interaction clarity can be lost (C9, G3). Workers may be forced to make sense of unfamiliar, strangely named structures that are essentially external manifestations of a product team's own misunderstandings. Central domain artifacts may be overlooked or underemphasized (A9), which may cause workers to see resulting applications as irrelevant (K3) and excessively effortful to learn (D2, D3).

See also: B, C, F, H, I, J, K1


Setting up a new clinical study in my lab's information management application means creating a set of expected and familiar items for my plans...
Clinical Scientist
These are the things that we talk about in our lab, that "live" in our lab's shared database...
OBJECTS CREATED FOR A SMALL CLINICAL STUDY
Study File
Automation Procedure
Clinical Samples
Test Tubes

Key *application envisioning* questions:

What artifacts do targeted knowledge workers currently focus on in the work practices that your team is striving to mediate, and how might these objects be embodied in your application concepts? What new interaction objects are implied in your sketches of functional possibilities?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What inventory of artifacts from targeted individuals' environments might your team consider as potential elements and references for your computing tool?

Who uses each type of artifact, and how do they use them? How does usage vary across targeted organizations?

What characteristics do workers value in the objects that they currently use? What emotional connections do they inspire?

Are these artifacts primarily physical, primarily digital, or a combination of the two? How permanent or malleable are they?

How have these artifacts evolved into their current state within particular organizations or larger professions? What can be learned from recent evolutionary steps in these historical trajectories?

What nomenclature do targeted workers from different organizations and market segments currently use in reference to specific artifacts?

Which existing objects might benefit from meaningful subdivision or elaboration within the setting of your team's application concepts?

How could useful representational characteristics of certain artifacts be preserved or even enhanced?

Which existing artifacts could be difficult to effectively translate into a cohesive and well resolved onscreen object? How might these challenges impact your team's sketched functionality concepts?

What conventional interaction objects, found in many computing tools, are implied in your ideas about mediating work?

How might your team invoke workers' valuable conceptions of known artifacts as part of new interactions and representational forms?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B2. Flexible Identification of Object Instances

In order to effectively support knowledge work practice, certain types of interaction objects typically need to have multiple instances. Especially for those object types that are higher volume and a main focus of ongoing effort, product teams can envision flexible, complimentary options that could allow workers to apply meaningful identification schemes.

Examples from three knowledge work domains:

> A financial trader often needs to find previously completed transactions in his trading application. He can identify individual trades by their unique trade numbers or a combination factors such as the security traded, the quantity traded, and which trader in his group completed the deal **(see illustration on next page)**.

> An architect names and saves a selected structural element as a reusable template within her building modeling application. She applies a variety of searchable attributes to the new template, including the building element's function and material composition.

> A scientist identifies a new clinical sample in her laboratory information management application using a code for the tissue's donor and the experimental treatment that it will undergo.

The identification of an individual artifact can trigger a knowledge worker's memories and understandings of its place and meaning in their work (A, D3). The naming or categorization of an artifact can also act as a bridge to existing, related information (B3).

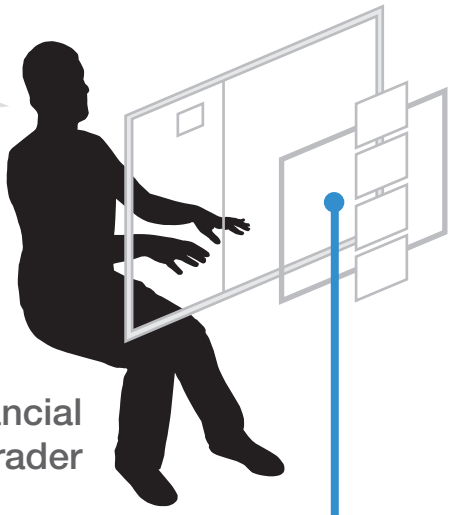
Product teams may find that identification requirements can vary drastically for different types of interaction objects in their application concepts. Granular objects, such as a single point in a drawing, often require no identification other than their location in space. Low volume objects based on domain artifacts may need only a simple, yet highly flexible, "name" field (A9) in order to be effectively integrated into workers' practice. High volume, persistent objects (I) that are a primary focus in work activities (F2) can require a number of complementary identification attributes (K). In situations where teams find it difficult to envision standardization of these attributes, knowledge workers may value customizable identification functionality (C8) that allows them to develop information management strategies (I1) to meet their local needs (A7, A8, K1).

When product teams do not actively consider how individuals and organizations could meaningfully identify various interaction objects, opportunities to facilitate important clarity within diverse work practices can be lost. Inadequate object ID information can hinder many aspects of knowledge work, such as retrieval of application content (I2, I3) or the orchestration of collaborative action (A7, C7, G4). When faced with limited object identification functionality, workers may define cumbersome and elaborate naming conventions in an effort to address a range of identification needs (D2, D3, E1, E2).

Conversely, excess identification fields and options may create situations where workers feel that they need to enter more data than is practically valuable.

See also: B, C5, F1, F11, G2, H4, I, J5, J6

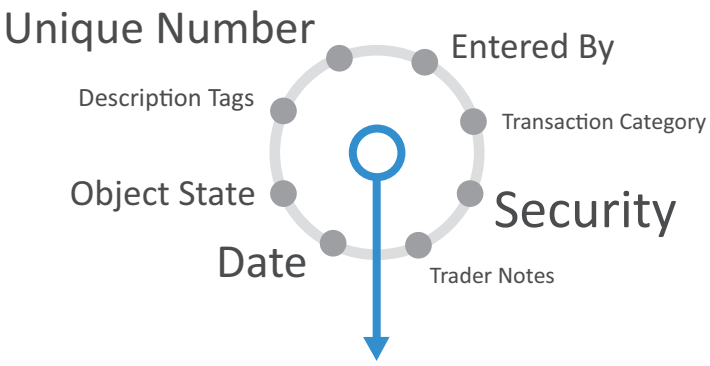
There are so many trades made even in an hour, it's hard to remember very much information about any one given deal...



Financial Trader

One thing that is very helpful is that I can search by entering any combination of different identifying aspects for a trade...

IDENTIFYING ATTRIBUTES



And eventually, I'll figure out a way to navigate the information in order to find a certain deal or whatever I'm looking for...

Key *application envisioning* questions:

What flexible, complimentary methods might your team envision to allow targeted knowledge workers to identify and easily recognize certain instances of interaction objects within your application concepts? How might different identification options drive different approaches to information structuring and seeking behaviors?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently identify specific instances of their workplace artifacts — especially those items that are involved in the tasks and larger activities that your team is striving to mediate?

Are existing methods based on free form names? Do they contain categorical identification attributes?

What important variations in identification approaches can your team find within and across targeted organizations?

How might your team translate existing identification methods into your application concepts? How could existing methods be extended?

What object identification information will satisfy the majority of cases? How much identification might be too much?

What customizations might your team envision to support uncommon object identification needs within targeted organizations? Will this functionality provide enough value to offset its added complexity?

How will workers enter object identification data in your sketched functionality concepts? What innovative methods might your team envision to valuably decrease these efforts?

As volumes of data build up over time, what secondary information could also serve as identification for different types of interaction objects? What implicit attributes could become elements of larger identification schemes?

How might your team's ideas about such schemes relate to your other design responses for supporting work in the context of volumes of information?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B3. Coupling of Application and Real World Objects

Some knowledge work applications contain interaction objects that are extensions of, rather than replacements for, offline artifacts. In these cases, product teams can envision interactions that tightly couple onscreen and off screen equivalents in order to promote a more efficient, direct, and unified experience.

Examples from three knowledge work domains:

> A scientist places a test tube containing a clinical sample into a rack next her computer workstation. Her lab's information management application reads a signal emitted from a small tag on the test tube, then displays stored information about the tube's contents on her screen **(see illustration on next page)**.
>
> An architect scans a cardboard model of a building form into her building modeling application. She gives the computing file the same name as the one she has written in black marker on the cardboard version.
>
> A financial trader scans a barcode on a paper trade cancellation form that was faxed to him. His trading application pulls up the associated trade and prompts him to initiate the cancellation process.

The adoption of computing into knowledge work practices typically does not mean that workers will suddenly switch to only manipulating symbols on screens. In many types of knowledge work, tangible, real world objects can remain an important part of individual or collaborative behaviors (A).

The coupling of offline objects to their digital equivalent, or other associated content within a computing tool (G6), can be considered a special type of coordination between representations of workplace information (F1). The experience of carefully designed, tightly coupled coordinations can extend both into and out of a computer's display. Physical objects can become interactive entry points into an application's content.

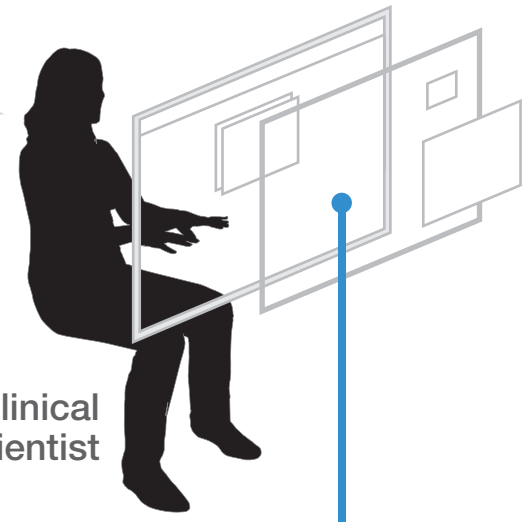
From the other side of the relationship, onscreen interactions can map back toward physical objects, potentially creating new forms of pervasive awareness and telepresence.

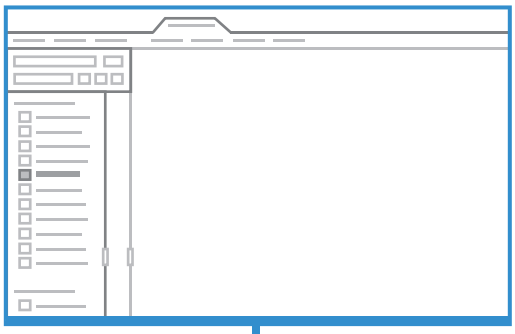
Product teams can envision compelling, goal oriented experiences of connective threads between the screen and material objects. At a minimum, common identifying information (B2) between tangible artifacts and their application equivalents can act as a coordinating link (G5). Some knowledge work domains present opportunities for teams to envision more extensive coordination of the physical and the intangible, based on, for example, well characterized transition points in work sequences (D5, G1, J3).

When product teams do not actively consider where bridges between physical and digital objects could be compelling, feasible, and valuable in their application concepts, opportunities to provide a powerful sense of direct action and engagement can be lost (K13). Workers may experience online and offline instances of an object as disjointed and separate, which can make such applications more effortful to use when compared with potential scenarios of interactive connection (D2, D3, K2, K6).

See also: B, E, F1, F9, G, H4, I5, J2, J5, J7, L3, M1

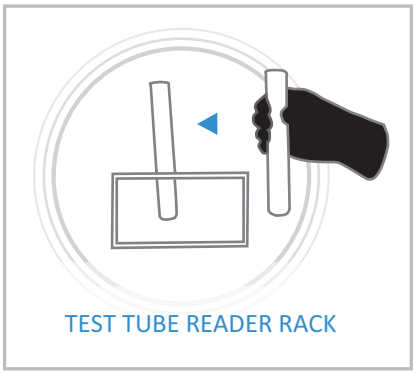
Our lab's information management soft-ware is set up to "know," in a limited way, where things are in the lab...





So, for example, right now the application has no data displayed...

I'm going to put a test tube into the reader rack, and it will pull up related data from the system...



And now the reader has found the test tube and brought the sample up onto the screen...

It shows related sample data because I'm in the samples view of the tool...

Key *application envisioning* questions:

What interaction objects in your team's application concepts could benefit from a preserved connection to related off screen artifacts? What functionality concepts might your team envision to allow targeted knowledge workers to usefully recognize and meaningfully act through these connections?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What real world objects in the work practices that your team is striving to mediate are not likely not be replaced by an onscreen equivalent?

What scenarios could potentially lead to new physical objects being created based on the contents of your product?

What types of targeted organizations might be more likely to "hold onto" the physical incarnations of their otherwise onscreen work? Why?

What targeted tasks or larger activities might benefit from the tandem use of both physical and digital instantiations of an artifact?

What coordinations between interaction objects and their off screen equivalents, such as matching identification information, could provide clarifying utility and reduce workers' efforts?

What larger technology and market trends could influence your team's ideas about intentionally coupling physical and digital objects? What might be feasible if the value proposition was compelling enough?

What valued characteristics of real world objects could be difficult to include in corresponding onscreen objects, and vice versa?

How might these deficiencies drive workers to turn to the "other" version of an object? How could these transitions be crystallized into goal directed interaction pathways within your application concepts?

What novel interaction methods might your team envision to tightly couple certain real world objects with associated content in your computing tool? How could these methods directly bridge well characterized seams in specific work practices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B4. Object Associations and User Defined Objects

Interaction objects can carry default and worker defined linkages to other objects within a computing application. Product teams can envision how clear and actionable presentations of these object associations could allow workers to offload effort while acting in informed and confident ways.

Examples from three knowledge work domains:

> An architect groups together a series of elements in her building modeling application and identifies it as a new type of window assembly. The newly grouped object maintains easily recognizable linkages to several important functional properties as well as her early notes on its proposed construction **(see illustration on next page)**.

> A scientist deletes a set of clinical samples from the scope of a specific report in her analysis application. The report dynamically updates with a notation that certain data has been removed from its contents and that the excluded data is still persistently available in the database.

> A financial trader uses his trading application to group together differing quantities of several different securities into a large deal proposal. He then divides the contents of the proposal into three different categories based on the estimated values of each line item.

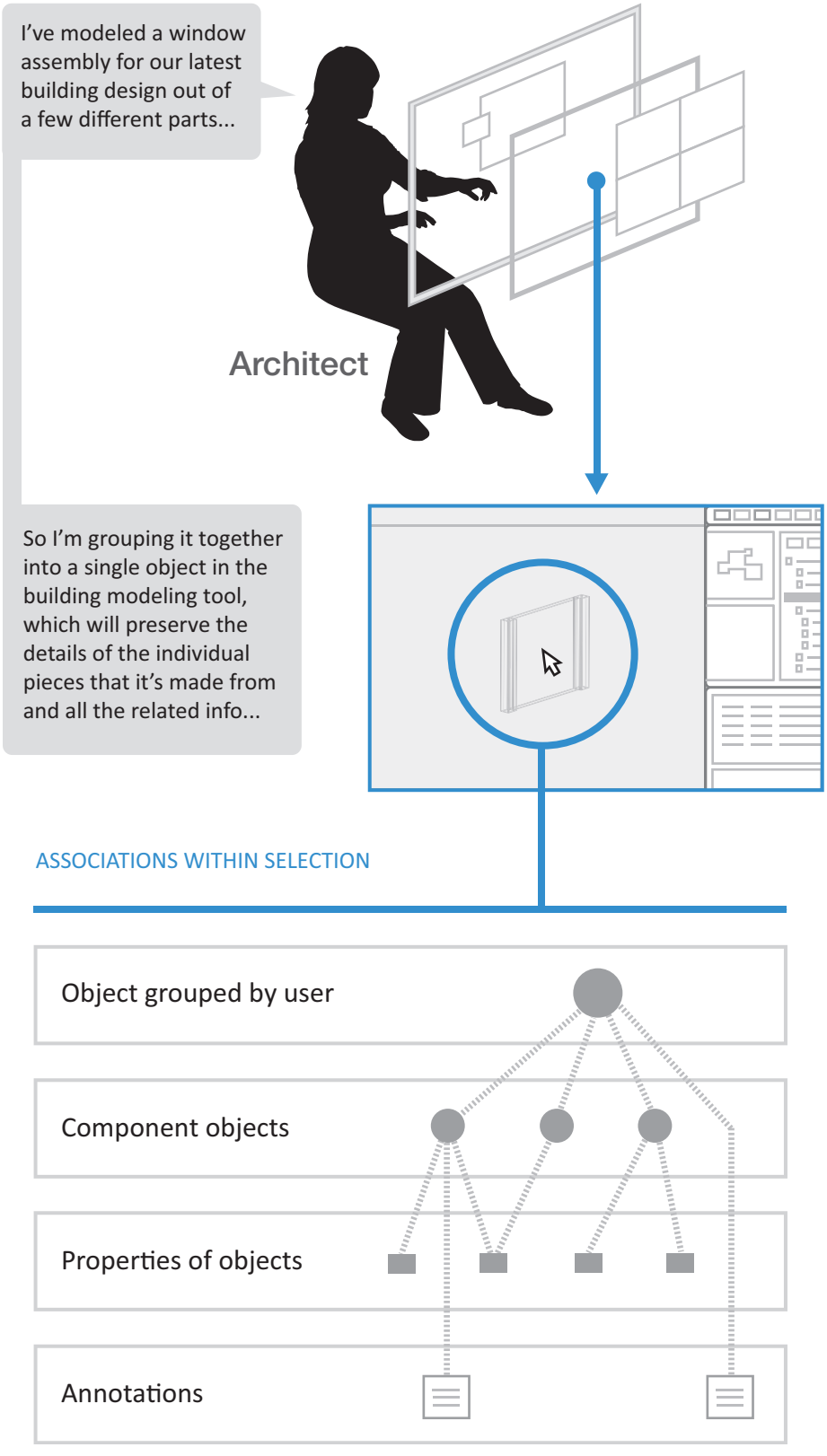
Knowledge workers create, manage, and make use of relationships in information. Computing applications can excel at storing, presenting, and acting through complex associations that their users would otherwise find difficult or nearly impossible to manage (E).

As part of envisioning interaction objects (B1) and their potential roles in work practices (A, B8, B9), product teams can map out inherent hierarchies and linkages that need to be made clear to users of their computing tools. Teams can also envision circumstances where it could be valuable to allow workers to define their own associations, either implicitly, through the attribution of similar traits across multiple objects, or explicitly, by associating selected elements to form larger structures.

Associations between objects can allow workers to usefully propagate a single interaction across a number of related elements. Clearly communicated conceptual models (C1) and visual representations for levels of selection (F, G2) can be essential for supporting different scopes of action. These factors can also be crucial in the context of collaborative, shared data environments (C7, G4).

When product teams do not actively consider the potential role of object associations within their application concepts, resulting products can contain serious flaws. Workers may commit critical errors when actions cascade unexpectedly through linkages that are difficult to trace and predict (C9, G3). When expected linkages are not present and cannot be created (G5, M4), workers may have to effortfully make individual modifications to objects in series, rather than acting on larger groupings (D2, D3). Absent cues about relationships between objects may also necessitate time consuming, trial and error exploration.

See also: B, C5, C8, F, G6, I, K3, K11


I've modeled a window assembly for our latest building design out of a few different parts...
Architect
So I'm grouping it together into a single object in the building modeling tool, which will preserve the details of the individual pieces that it's made from and all the related info...
ASSOCIATIONS WITHIN SELECTION
Object grouped by user
Component objects
Properties of objects
Annotations

Key *application envisioning* questions:

What connections and interrelations could be present in the inventories of interaction objects that your team has identified? How might your sketched functionality concepts allow targeted knowledge workers to define, recognize, make senses of, navigate, use, or even defend against these associations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What linkages between artifacts do targeted individuals currently manage in the tasks and larger activities that your team is striving to mediate?

How do people think about these relationships? What nomenclature do they currently use to describe different associations and connections between artifacts?

What default linkages and hierarchies of interaction objects are implied within your team's application concepts?

What implicit associations between objects might be created through the attribution of similar traits across multiple object instances?

In what scenario contexts might it be valuable to allow workers to group selected interaction objects together into larger structures?

What goal directed pathways of action could be made available based on the presence or absence of certain object associations?

How might certain user selections and actions trace through the linkages that your team has envisioned?

What conventions might you apply throughout your product to promote consistent and understandable behaviors in object relationships?

How might clear conceptual models for different types of object linkages be communicated within your functionality concepts?

How could legible design communication prevent unexpected effects via unseen connections?

How might your team's ideas about object associations inspire you to ideate valuable new interactions and representational forms?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B5. Object States and Activity Flow Visibility

Understanding the current state of interaction objects can be crucial for the effective planning and execution of knowledge work. Especially for those object types that are higher volume and a main focus of workers' ongoing efforts, product teams can envision appropriate states that could communicate potent meaning and directive pathways of action.

Examples from three knowledge work domains:

> A financial trader reviews the status of a number of negotiation messages in his trading application to determine whether he needs to put any more effort into them. Scanning the list, he decides to move forward with booking other trades **(see illustration on next page)**.
>
> An architect waits for her building modeling application to render one segment of a complex design. Since all of the elements involved in that rendering are shown as locked for editing until the process is complete, she temporarily navigates to another area in the model to make edits.
>
> A scientist turns to her lab's information management application to review how many samples in a clinical study have not yet been processed. This information allows her to estimate a timeframe for the study's completion and to plan her lab technicians' work schedules.

Object states can be displayed implicitly, based on various object attributes, or explicitly, through preordained state indicators. Recognizable and meaningful states can become an effective basis for organizing (I1) and locating useful categories of application content (I2, I3). They can also determine which objects and associated avenues of interaction (C4) are visible to particular users at a given time (C5). These gleanings can allow knowledge workers to prioritize their efforts and plan appropriate courses of action (D3, D5) in cooperative scenarios (A7, C7, G4) and standardized processes (C6, J3).

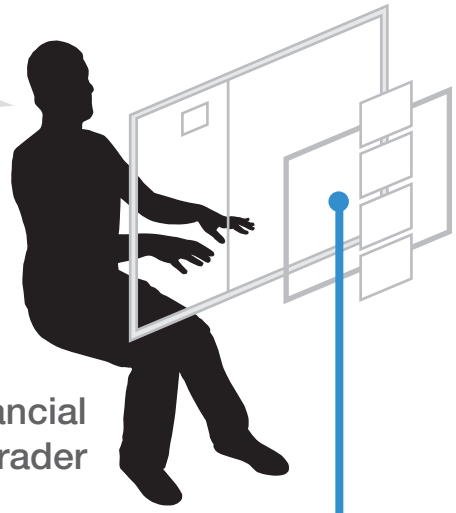
Product teams can clearly define appropriate object states based on their ideas about how different interaction objects might fit into mediated work. They can envision how these states could be communicated though domain language (F10) and other methods that invoke workers' deep seated understandings of place and priority (C1). Teams can also explore flexibilities that might allow organizations to define their own object states to meet local needs (C8, K11).

When product teams do not actively consider the potential role of meaningful object states in their application concepts, opportunities to clarify knowledge workers' current progress and options can be lost. When explicit state information is unclear or excluded, workers may need to effortfully dive into the attributes of interaction objects in order to derive their status (D2). These deficiencies may also lead to errors in timing (C9, G3) and less optimal work outcomes (L1).

Conversely, object states that push too much standardization can lead to confusing and dissatisfactory limitations on work processes (A9). These design issues can force individuals and organizations to adopt unwanted changes in their cultures in order to match a system's seemingly arbitrary rules (A4).

See also: A, B, C10, E3, F, J1, H, I7, M1, M4

I'm negotiating a bunch of tougher deals at the same time, so I'm constantly going back to my messages to see what I need to respond to...



Financial Trader

All of the messages have easy to understand codes that tell me the state of each negotiation...

MESSAGES BY STATE CATEGORIES

Negotiation with Minor Changes	Negotiation with Major Changes	Cancelled Negotiation
1 Message	2 Messages	4 Messages

It looks like things aren't going so well with most of these, but I'll wrap up the one with minor changes before moving on to making new deals...

Key *application envisioning* questions:

What useful or necessary states can your team envision for key interaction objects in your application concepts? How might these object states play meaningful and directive roles in your functional responses for targeted knowledge work practices?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently categorize the states of different artifacts in the tasks and larger activities that your team is striving to mediate?

How do the physical placements and observed "ownership" of certain artifacts currently imply state information?

What do particular states "say" about the work that has been accomplished on or around an artifact? The work that needs to be done? The people involved?

What differences can your team find in how targeted organizations categorize these states? What differences may be difficult to reconcile?

Will the states that workers currently talk about and use translate well into an interactive application? Why or why not?

What novel states might targeted individuals and organizations value? How might the introduction of a new computing tool present opportunities to usefully standardize certain categorizations in work process?

What new states will your team need to introduce in order to clarify and support your functionality concepts? What design communication could effectively explain these new conditions?

How might new states offload the need to be vigilant for certain changes in interaction objects, potentially tying into alerting functionalities?

Has your team envisioned any single track processes that must be completed without interruption in order to be effective? How might these "untouchable" intervals influence objects' states?

How might error prevention and handling scenarios require additional object states? Could these error states impact larger, application states?

Which pathways of action might be enabled or disabled when an interaction object is in various states?

Where might certain object state categorizations prove to be too confining for open or variable work practices?

What interactions and visual representations could allow users to usefully understand states across collections of similar objects?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B6. Flagged Variability within or between Objects

There are often aspects of interaction objects, outside of any explicit states, that are important to call to knowledge workers' attentions in certain contexts. Product teams can envision how adaptive flagging of central variabilities could reduce the effort needed to examine key characteristics of individual objects.

Examples from three knowledge work domains:

> A scientist notices that her lab's information management application has flagged a sample as having different attributes than other the samples that are associated with the same clinical research participant. She reviews the flagged sample's details and discovers that a technician has made a data entry error that she needs to investigate before proceeding **(see illustration on next page)**.
>
> A financial trader reviews a list of incoming deal proposals. He looks for items that his trading application has automatically flagged as being immediately executable, indicating that his firm has sufficient quantity of the holding in question to meet a proposed deal's stated needs.
>
> An architect switches to a view in her building modeling application that flags elements of the project's 3D model where her team has not applied any project requirements tags.

When an abundance of interaction objects are displayed simultaneously, knowledge workers' content rich computing displays may turn into dense, perceptually "flat" sheets of information (B1, I). While defined object states (B6) can determine available avenues of interaction (C4) and other important factors, secondary, strictly informational characterization of interaction objects can help workers attend to and make sense of important information (D3, D6, F10).

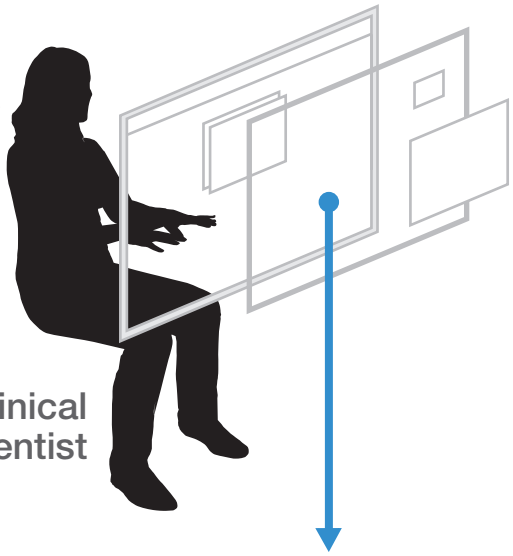
Visible and meaningful flags that indicate certain categorical conditions in an object's attributes (A4) can be useful while workers organize (I1), retrieve (I2), browse (F5), and transform application content (F8) to meet particular goals. The categorical basis of a flag can come from specific information that workers' have previously entered, or it can be a derived value that is automatically calculated based on domain appropriate rules (C8, E3, E4).

Certain flags may appear only in the context of certain interactions or object states (B8, B9). For example, they can be implemented as a method for preventing human error in defined processes (C9, G3), calling out values that could be important for effective decision making.

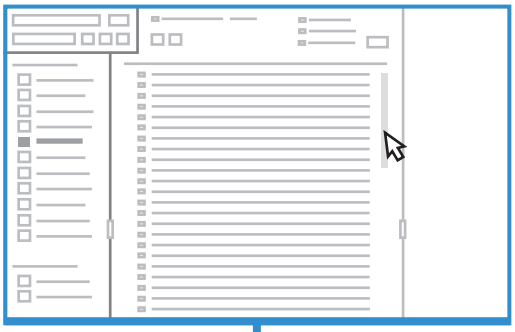
When product teams do not actively consider the potential role of flagged variabilities for key objects in their application concepts, knowledge workers may need to dive into the details of complex data structures in order to investigate their attributes. Due to constrained time and attention, people may not always perform these additional efforts (D2), potentially leading to crucial errors or important losses of insight that may negatively impact work outcomes (L1). In the absence of informative flags, filtering and sorting on specific object attributes (I3) may provide more active pathways to accomplish similar goals.

See also: A, B, D, F, G4, G6, H3, J1, L, M1

I'm using our lab's main data management application to find all of the samples that are involved with our current clinical study...



And I've got a set of search results back that I'm scrolling through...



And the tool is saying that there is something wrong with one of the samples...

It looks like a treatment is being applied that is not found anywhere else in this study...

That's a simple data entry error from earlier that I can fix right now...

Key *application envisioning* questions:

Beyond defined states, what specific pieces of information about interaction objects might be especially interesting or useful to targeted knowledge workers during the course of their practices? How might your team informatively communicate these key variabilities through perceptually salient cues?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What characteristics and signs about artifacts do targeted individuals currently gravitate to in the tasks and larger activities that your team is striving to mediate?

Why are these characteristics and signs useful in certain work practices? What decisions and actions do they inform?

How might these existing attributes be translated into valuable flags for your team's envisioned interaction objects?

What meaningful new flags might you envision to call out key object information within the interactive flows of your team's sketched functionality concepts?

How could informative flags prevent human error in certain activity contexts? What specific information about an interaction object, if left unknown, might lead users to act in error?

What might the aesthetic presentation of "flags" look like in certain functionality concepts? How might these cues relate to any error management conventions your team has defined?

How could certain flags convey their priority, simply based on their level of salience on the screen?

How might your team's ideas for specific flags inspire you to ideate valuable new interactions and representational forms?

What flagging categories and conventions might your team establish and consistently apply throughout your computing tool?

Could simplifying certain interaction objects make more sense than flagging important pieces of information within them?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B7. Object Ownership and Availability Rules

Similar to offline, real world artifacts in a knowledge workplace, onscreen interaction objects can benefit from clear and consistent rules governing who can perform actions on or with them at a given time. Product teams can envision and communicate rules that are culturally appropriate, logically feasible, and understandably clear.

Examples from three knowledge work domains:

> An architect selects a segment of a design in her building modeling application to "check it out" so that she can make some small modifications, only to find that the segment is currently checked out by a consulting civil engineer. The application provides her an option to work in an alternate version of the segment, which no one else will be able to access until she later merges it back with the main version of the model **(see illustration on next page)**.

> A financial trader attempts to trade all of his firm's holdings of a particular security, but his trading application displays a message that prevents him from completing the transaction. It seems that part of the total amount has been locked by another trader who has indicated that he wants to use it as part of a higher value deal.

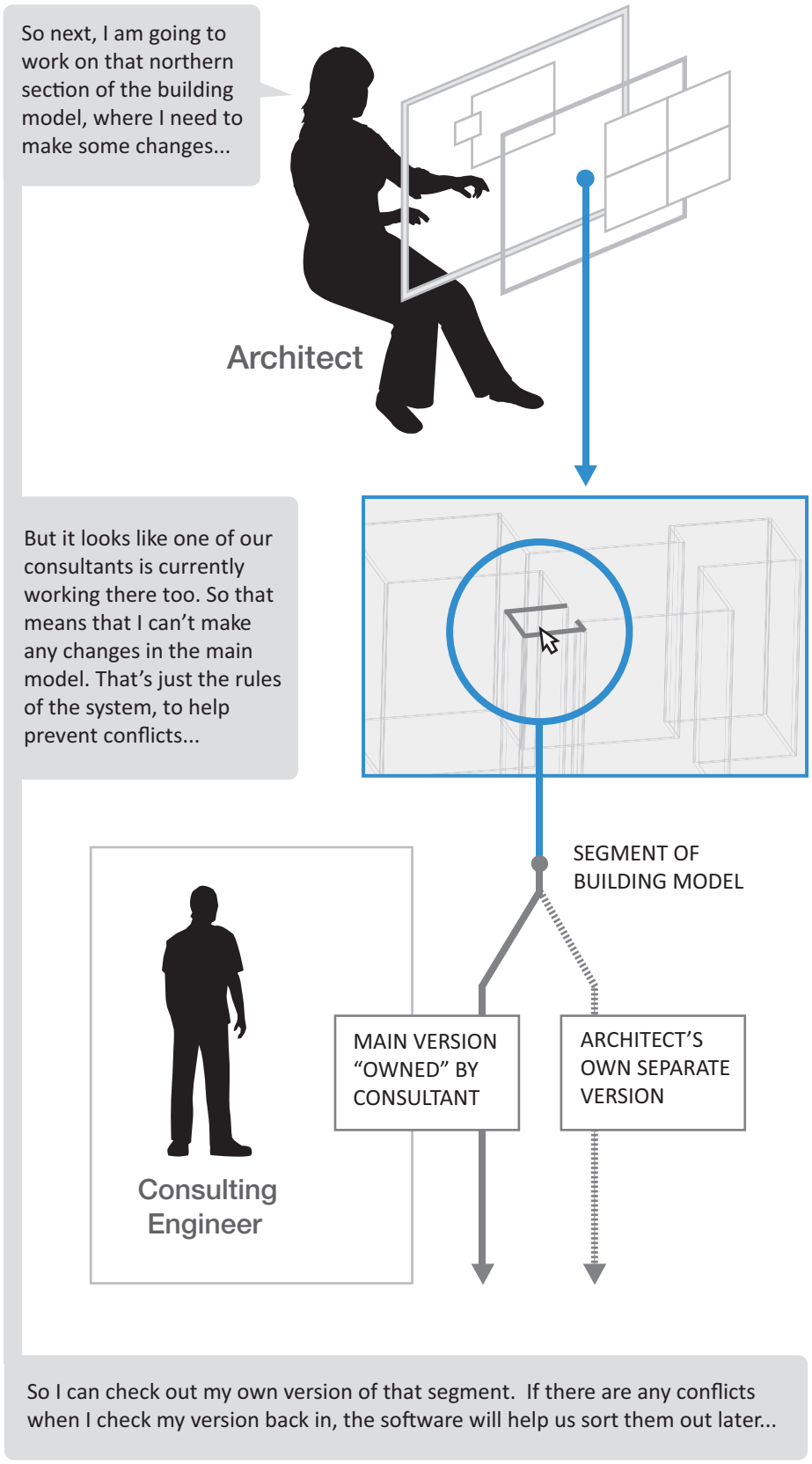
> A scientist cannot change the name of a clinical sample in her lab's information management application because an automated instrument is currently processing portions of the sample's tissue.

In order to effectively accomplish work in their complex cultural and organizational environments (A1), knowledge workers often become skilled at cooperatively managing access to and use of shared information, tools, and other artifacts. In the inherent abstraction of shared computing environments, workers may find it difficult to hold onto some of these existing skills as portions of their material culture, and its associated practices, are migrated toward individualistic computer screens. Workers can become somewhat dependant on their applications for clarification around who currently "owns" what and how they might use it themselves (C7, G4).

Applications can support both division of labor (J3) and collaborative practices (A7, J4) by reinforcing understandable rules for the ownership and availability of objects (C1). Product teams can envision these rules based on contemporary conventions, which may vary for different types or levels of objects in a system (C3). Availability rules can be tied to user permissions (A2, C5), though such rules can also be found in applications without any notion of differential user privileges. Products that will not have features for controlling object ownership can sometimes be envisioned to leverage available rules from a coordinated storage technology, such as a file server or database (K10).

When product teams do not actively consider how they might clarify object ownership and availability rules, resulting applications may present opportunities for frustrating conflicts and versioning problems in collaborative and cooperative scenarios (H1). Without a clear, consistently applied model of when objects are accessible for intended actions, workers may find planning the flow of their practices to be excessively effortful, inaccurate, and unpredictable (D2, D3, D4).

See also: A, B, D6, E3, F10, H, I, J1, J5


So next, I am going to work on that northern section of the building model, where I need to make some changes...
Architect
But it looks like one of our consultants is currently working there too. So that means that I can't make any changes in the main model. That's just the rules of the system, to help prevent conflicts...
SEGMENT OF BUILDING MODEL
MAIN VERSION "OWNED" BY CONSULTANT
ARCHITECT'S OWN SEPARATE VERSION
Consulting Engineer
So I can check out my own version of that segment. If there are any conflicts when I check my version back in, the software will help us sort them out later...

Key *application envisioning* questions:

Based on your team's understanding of targeted cultural environments and knowledge work practices, what rules can you envision for key interaction objects to ensure that they are "owned" and accessed by workers in appropriate and useful ways?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What rules do targeted individuals implicitly or explicitly follow to promote the effective sharing of artifacts within the work practices that your team is striving to mediate?

How much emphasis do workers place on ownership and availability rules in their observed practices? Are these rules valued and useful components of their operative cultures?

How, specifically, do existing rules promote coordination and prevent conflicts? What would happen if they were not in place?

How could your team translate these existing practices into ownership and availability rules for interaction objects in your computing tool?

What new opportunities for conflict might your product create, simply by bringing collaboration and cooperation into an abstract computing environment?

What larger design and technology trends could influence your team's ideas about what appropriately applied, logically feasible, and understandably clear rules could look like?

What conventional patterns might your team reference in the design of object ownership and availability interactions?

What overall, "global" rules might your application concepts follow in order to control the ownership and availability of objects?

Which of your team's envisioned scenarios for mediating work could present unusual situations where conflicts may occur and additional rules could be valuable? How might these special solutions differ from your "global" rules?

How could users' interactions within your sketched functionality concepts help them to situationally build accurate conceptual models around object ownership and availability in your product?

How might these rules relate to your team's ideas about workspace awareness at the application level or within specific functional areas?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B8. Explicit Mapping of Objects to Work Mediation

Even though a general understanding of an interaction object can carry with it expectations of certain related actions in a knowledge work application, product teams can prevent oversights and drive interaction clarity by explicitly mapping how important objects could fit into targeted operations, tasks, and larger activities.

Examples from three knowledge work domains:

> A financial trader appreciates how his trading application gives him the shortest set of available action options when he looks at a trading message. He feels that these targeted options, which are based on each message's current state, allow him to work faster, without second guessing what else he could be doing **(see illustration on next page)**.
>
> A scientist likes that the interface of her new analysis application provides a range of choices for exploring data at various levels of aggregation, including actions that used to be "missing" or "hidden" when she made certain selections within other visualization tools.
>
> An architect becomes accustomed to the actions that are available to her in a new building modeling application. Even though it is an open workspace tool, with many options available for her to select at any given time, it feels like the functions that she ends up seeing first often correspond to what she is presently currently trying to accomplish.

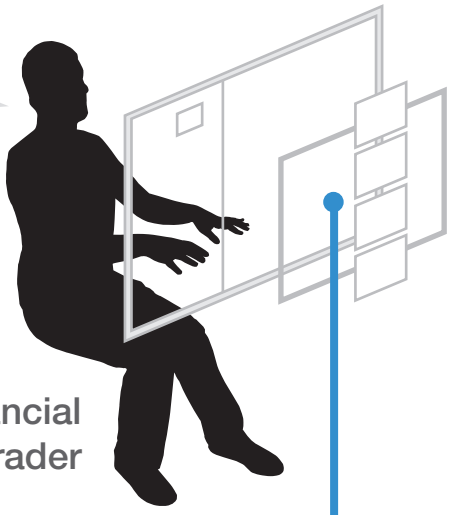
Knowledge work applications are designed to support specific interactions, and they often do not have the plasticity to be applied outside of that range of intended use. Product teams can envision the narrative mapping (G1) between onscreen subjects and corresponding options for action as the intersection of specific types of interaction objects (B1), their current state (B5), the current state of the application (C10), and the work practices that an application is being designed to mediate (A).

A separate model of what workers will want or need to do with key interaction objects can provide product teams with a strong foundation for envisioning an application framework (C), including interaction pathways (C4), appropriate and consistent interaction patterns (C3, K6), support for collaboration (B7, C7, G4, J4), and support for explicit division of labor (J3, G5). Since a single object can be tied to drastically different tasks or larger activities (A5), teams can use an overall map of each object's potential actions to envision tailored design responses that could match workers' goals in each circumstance (A7, A8).

When product teams do not actively consider how important interaction objects might map to the breadth of work that they are striving to mediate with their application concepts, resulting tools may contain oversights in definition and design that make them difficult or impossible to use in some activity contexts (A6, D2, D3, D4). While these issues can often be addressed through iterative corrections, cohesive design of activity oriented wayfinding often involves something more than the sum of smaller, cumulative changes.

See also: B, F, G, H, J1, K, M1

Things in this trading tool have concise, intelligent action lists that help me make faster decisions...



For example, depending on the state of a trade form, I only get options to act that make sense given that state...

AVAILABLE OPTIONS BY STATE CATEGORIES

Blank Trade Form	Negotiation with Major Changes	Cancelled Negotiation
		
4 Action Options	3 Action Options	1 Action Option

I don't want to see any buttons for options that I can't click...

The people that made this tool understand how we trade, and so I don't have to think about those little things...

Key *application envisioning* questions:

How, specifically, could the interaction objects that your team has envisioned fit into the knowledge work operations, tasks, and larger activities that you are striving to mediate with your application concepts? What important relationships between objects and actions might you be overlooking?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How are certain artifacts currently used in different activity contexts? Are these usages consistent across targeted organizations?

How might your team map your emerging ideas about mediating knowledge work as the actions that can be performed on different types of interaction objects?

What potential actions could be most important for key object types? The least important?

Where could the nature of a particular action vary based on the type of interaction object that it is performed on? How might your sketched concepts reflect these differences?

Which objects typically serve as tools for acting on other objects, rather than being the recipients of actions themselves?

How could the states of interaction objects, or larger application states, influence the actions that are available at a given time?

How might the sum of your team's object and action mappings inform your ideation about application frameworks and interaction pathways?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B9. Common Management Actions for Objects

Some types of interaction objects in computing applications will typically require a conventional set of management actions, such as create, copy, edit, and delete. Product teams can map available management actions for different types of interaction objects, envisioning what common functionalities might look like in different object contexts.

Examples from three knowledge work domains:

> An architect is refining an early design exploration in her building modeling application. She creates a building element, copies it, and pastes a duplicate element in another part of the building design. She likes how easy it is to create repetitions within the model, allowing her to quickly visualize her ideas for building form **(see illustration on next page)**.
>
> A scientist deletes a file in her analysis application where she had pursued the wrong approach to a clinical research problem. By deleting the outputs of her faulty exploration, she ensures that it will not be mistakenly accessed by herself or others in her lab.
>
> A financial trader creates a new categorizing attribute in his trading application to supplement the product's defaults. From that point on, he and other traders in his group will have the option of tagging the new informational attribute onto their pending and completed deals.

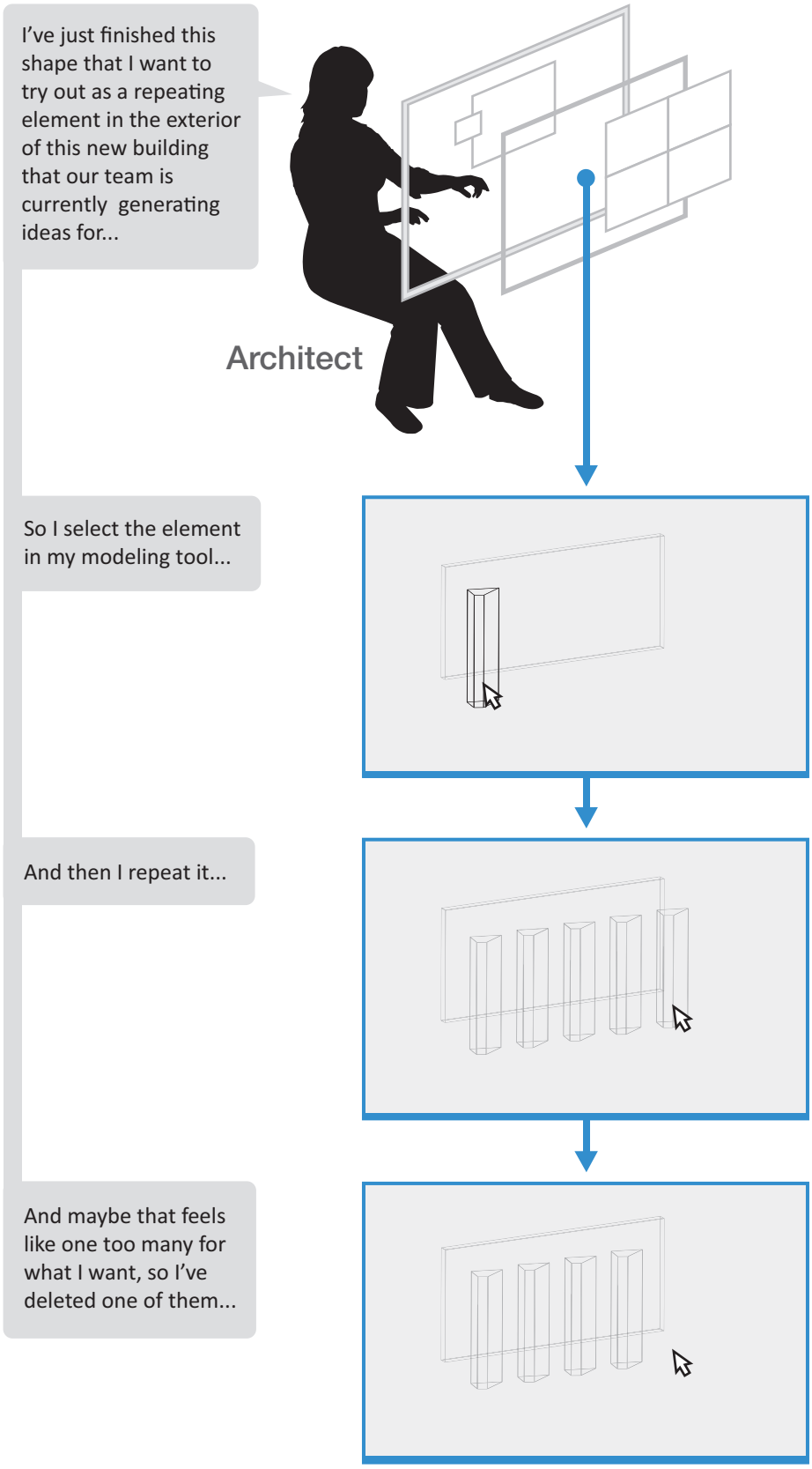
Interactive applications typically need to provide knowledge workers with some standard actions for working with multiple instances of onscreen objects (B1). Whether the interaction object in question is an overall file or a much smaller element, the clarity and directness of "management" actions can be an important usability concern.

To ensure that these interactions are sufficient and coherent, product teams can map the breadth of object management scenarios presented by their application concepts (B8, C3, G2). Management actions frequently include create, copy (E3, E4), edit, and delete. Other actions, like creating from a template (B10) or resetting to defaults (C8), can be usefully considered as "standard" for some types of objects (A4). In collaborative computing environments (A7, C7, G4), the presence or absence of management actions may depend on dynamic rules that prevent negative impacts on the activities of other workers (C5, J4).

When product teams do not actively consider object management actions in their application concepts, they can easily overlook central requirements due to their shared assumptions about what will be defined and implemented. When these oversights are present, users may, for example, be forced to view out of date content that they cannot delete (D4, I). To overcome some object management deficiencies, knowledge workers may need to develop and repeatedly enact excessively effortful work arounds (D2, D3).

Conversely, for some types of interaction objects, providing certain management actions, or even the ability to create multiple instances, may not provide sufficient value to warrant the additional complexity (A9, K6).

See also: A, B, C3, C4, H1, H2, I1, I2


I've just finished this shape that I want to try out as a repeating element in the exterior of this new building that our team is currently generating ideas for...
Architect
So I select the element in my modeling tool...
And then I repeat it...
And maybe that feels like one too many for what I want, so I've deleted one of them...

Key *application envisioning* questions:

What common management actions, such as create, copy, edit, and delete, could the interaction objects in your team's application concepts require or benefit from? What important management actions might you be overlooking?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently "manage" artifacts, in the computing sense of the word, while performing the work practices that your team is striving to mediate?

Which of the interaction objects in your sketched application concepts will have multiple instances, requiring some range of management actions? Which will not?

How might the understood "location" of individual objects factor into usable object management interactions? When is it not an issue?

Which object types should users not be able to delete? Why?

Beyond the basic set of management actions, what other actions could become standards within your computing tool, such as creating objects from a template or resetting objects' attributes to their default values?

How might your functionality concepts for managing different types of objects retain clarifying and learnable similarities? What important differences could be useful for managing particular object types?

How could object management functionalities provide strong feedback about action outcomes? What novel interactions and cues could reinforce the successful completion of these important tasks?

How might rules supporting collaborative practices influence the range and availability of object management actions?

What error prevention and handling concepts can your team envision to prevent loss of valuable information during object management actions?

How could associations between interaction objects be impacted by certain management interactions? When might workers benefit from understanding the lineages of these evolving associations?

What options might allow targeted organizations to appropriately set up different user permissions for managing different types of interaction objects?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

B10. Object Templates

When knowledge workers repeatedly generate instances of interaction objects with similar attributes, they may value the ability to create new objects from standard "molds." Product teams can envision functionality concepts that could allow workers to offload tedious data entry effort by tailoring and making use of object templates.

Examples from three knowledge work domains:

> A scientist creates files for a series of samples in her lab's information management application based on a template of attributes that she will use throughout a clinical study. She knows that these attribute consistencies will remove problems downstream, when her lab's technicians have actually run these experiments and she wants to analyze the resulting collection of data in her analysis application **(see illustration on next page)**.

> An architect uses her building modeling application to create a series of templated objects that satisfy some difficult project requirements. Creating these templates in advance will reduce work later, when these standard objects will appear repeatedly throughout the building's design.

> A financial trader exporting data from his trading application chooses from a menu of templates for different export content and layout formats. The ability to set up standard export options saves him a lot of time, especially when compared to manually reformatting each export file.

In many organizations, knowledge workers' activities involve, or even revolve around, the creation and recreation of specific work products or their constituent elements (A, B1). To support these tasks, product teams can envision template functionality that could allow workers to eliminate some of the effort needed to create certain interaction objects (E3, E4), while driving valuable standardization that can promote higher quality work outcomes (A4, C6, L1).

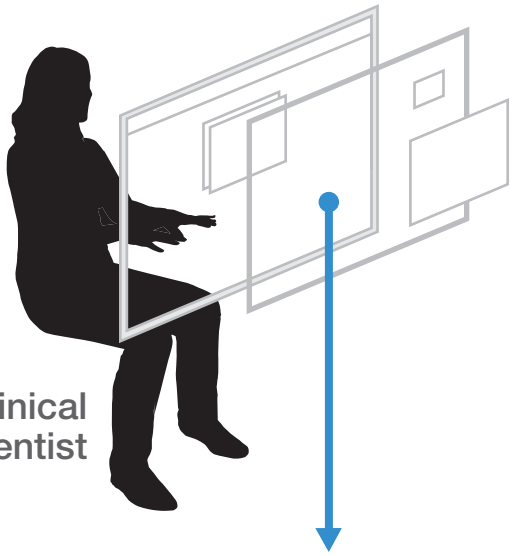
Product teams can envision templates as interaction objects in and of themselves, either provided as product defaults or flexibly created by workers to meet the needs of certain situations and practices (A7, A8, C8). Creating an object from a template can require subsequent effort to make the resulting object complete or to shape it to the requirements of a worker's current goals. Even when templates can be easily modified (K11), workers may need to edit or extend the attributes of resulting object instances.

When product teams do not actively consider the potential role of object templates in their application concepts, workers may find the process of creating essential interaction objects in resulting products to be excessively effortful (D2, D3). Additionally, key opportunities to provide beneficial types of standardization, which computers can excel at promoting, may also be lost (C9, G3, J6).

Conversely, for many types of interaction objects, template functionality may not provide sufficient value to warrant its additional complexity (A9, D4, K6). In some cases, the ability to create a copy of an existing object can suffice as a method for accomplishing the same goal.

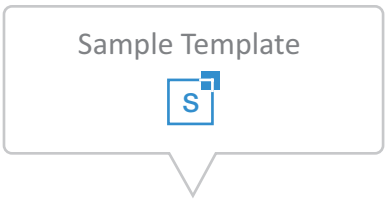
See also: A, B, C4, C5, H1, I1, K10, M1, M4

Consistency and information quality is incredibly important in research work, especially as volumes of data increase exponentially...

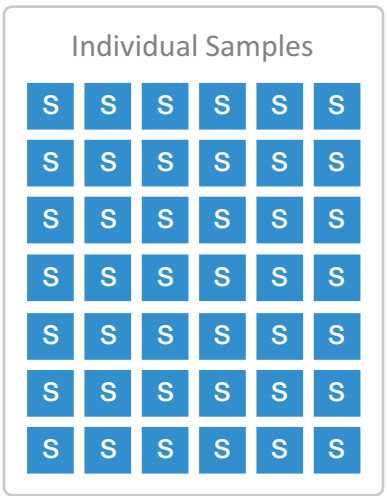


So when I've got a plan for a study and I'm creating an extended series of samples in the system...

I can create one sample template...



And then generate many individual, consistent samples in the software that are slight variations on that template...



Key *application envisioning* questions:

Where might object templates valuably decrease the effort needed to create common classes of complex information structures in your team's application concepts? What functional options could allow targeted knowledge workers to define, share, modify, and use these templates?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Where do targeted individuals currently take steps to promote valued sameness in artifacts as part of the work practices that your team is striving to mediate?

How do workers categorize and name common variants of workplace artifacts? How do these classification schemes vary across targeted organizations?

What value do these consistencies and categories provide?

Which "primary" interaction objects in your team's application concepts could become high volume and the repeated focus of work?

Which object creation tasks could potentially burden workers with data entry efforts?

How might templates be clearly and visibly differentiated from the type of interaction objects that they serve as a "mold" for?

Which of an object's information attributes should probably not be incorporated into a template's standardizing influence?

How might additional data entry effort, after an object has been created from a template, be clarified or lessened?

What pathways of action could flow out of the creation of an object from a template?

How might users trace an individual interaction object back to the template that it was created from?

What functionality concepts can your team envision to allow workers to clearly manage their various templates as specific workplace needs evolve over time?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C.

Establishing an Application Framework

Valued computing tools can tame complexity by structuring workers' interactions within comprehensible, consistent, and cohesive overall frames.

Designing such a clear organization requires deliberate and critical exploration of an on-screen tool's potential "shape" and "routes."

During *application envisioning*, product teams can synthesize common structural needs with their own resonating design ideas in order to sketch guiding models and larger interaction approaches for their products.

Early ideation about these application structures can "set the stage" for teams' evolving functionality concepts by both shaping and reflecting divergent ideas about potential user experiences.

Once product teams have generated a critical mass of sketched ideas about their application's potential roles in work practice, they can begin to meaningfully envision appropriate concepts for their product's high level "form." This overall form can communicate how a tool will basically work, and it can inherently define a framing range of useful interaction constraints. Teams can use these foundations to reshape their envisioning of key scenarios for mediated work, driving top down, systemic consistency and a larger design strategy across their proposed functional areas.

Sketching concepts for an application's framework does not entail exacting definition or design. Instead, it involves working through important constraints with only as much detail as is necessary to realize and communicate potential design concepts. Although many of these important constraints can arise organically from the ideation process, teams can also derive key constraints for their application's framework from well characterized challenges that are often manifested in computing tools for knowledge work.

This category contains 10 of the 100 *application envisioning* ideas in this book:

C1. Intentional and articulated conceptual models

C2. Application interaction model

C3. Levels of interaction patterns

C4. Pathways for task and activity based wayfinding

C5. Permissions and views tailored to workers' identities

C6. Standardized application workflows

C7. Structural support of workspace awareness

C8. Defaults, customization, and automated tailoring

C9. Error prevention and handling conventions

C10. Predictable application states

Product teams can use these ideas to explore notions of how their product could frame — both conceptually and in a literal interaction design sense — the knowledge work practices that they are striving to mediate. Although it can be safely assumed that early ideas about an application's framework will grow and change during the envisioning process and throughout product implementation, teams can deliberately preserve the essential character of the framing form that they have chosen to pursue.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Defining interaction objects" (B), "Clarifying central interactions" (G), and "Aiming for aesthetic user experiences" (L) categories.

C1. Intentional and Articulated Conceptual Models

Knowledge workers develop particular understandings of which work practices an interactive application is designed to support, how it essentially "works," and how it might fit into their own activities. Product teams can communicate their computing tool's intended conceptual models through application design and other channels.

Examples from three knowledge work domains:

> An architect finds that her studio's new building modeling application requires a substantially different mindset. Instead of drawing individual elevations, plans, and details for a project, her team will collaboratively create a single, shared 3D model of their building design. The new tool itself communicates this overriding distinction in numerous ways, including how various functions are named **(see illustration on next page)**.

> A scientist quickly develops a working understanding of how her analysis application calculates and presents certain values. Overall, the way the tool displays her lab's clinical data reminds her of a powerful, zooming microscope.

> A financial trader receives periodic updates from the vendor that created his firm's trading application. In one of these updates, he learns that he will need to develop a clear understanding of how, when, and why some new functionalities will usefully automate certain trade parameters.

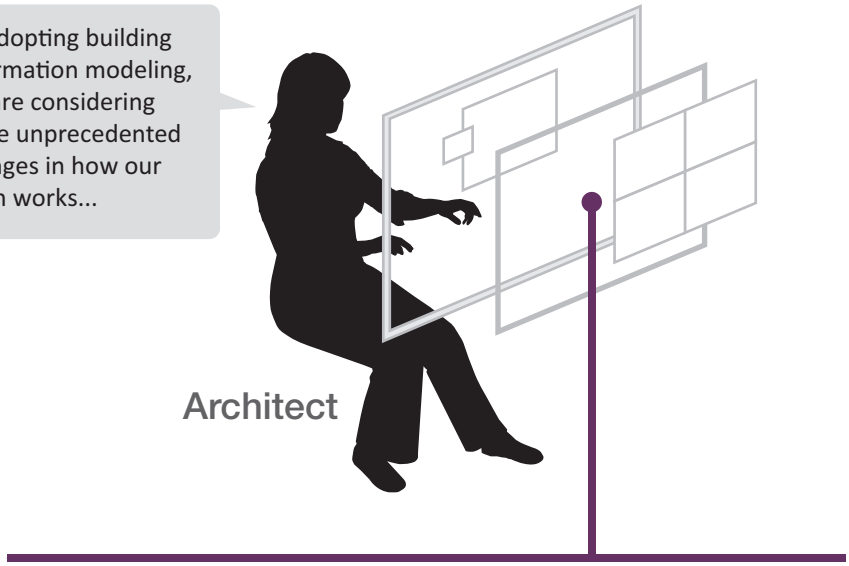
While mechanical tools can implicitly communicate how they work based on their construction, digital tools must be designed to communicate their purpose, offerings, and behaviors. Knowledge workers incorporate new technologies into their practices based on unfolding understandings of how available tools operate (A, K2, K6). Even though individual users develop their own conceptual models of a tool over time, product teams can attempt to shape these understandings by developing target models and striving to communicate them in their application designs.

To create compelling functional gestalts, product teams can envision conceptual models for their products that are framed by and build upon analogies and idioms known by their targeted audiences (C3, L2). Innovative models that simply and coherently present predictable relationships can also be quite successful (F3). Complex applications can contain multiple levels of nesting conceptual models, ranging from the overarching product framework, to individual functional areas (C6, B1), to functional variations driven by differing permissions and identity tailored views (A2, C5).

When product teams do not actively consider how proposed conceptual models could shape workers' experiences, opportunities to drive useful understandings of how an application essentially works can be lost. In the absence of clearly communicated conceptual models, people may experience computing tools as arbitrary collections of controls and pathways (K3), developing their own murky assemblies of functional interpretation. Even though these tools can be learnable after committed effort or training (D2, D3, M2, K7), potentially valuable functionality may remain undiscovered, misunderstood, or misused, requiring additional defensive measures to prevent errors (C9, G3).

See also: B4, C, E5, G1, L3, M1

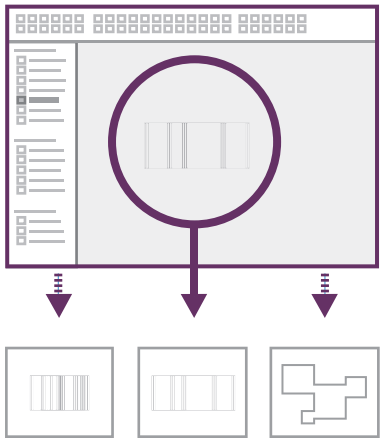
By adopting building information modeling, we are considering some unprecedented changes in how our team works...



OLD: CREATE ISOLATED DRAWING

The entirety of a building design is thought of as the sum of a set of separate architectural drawings.

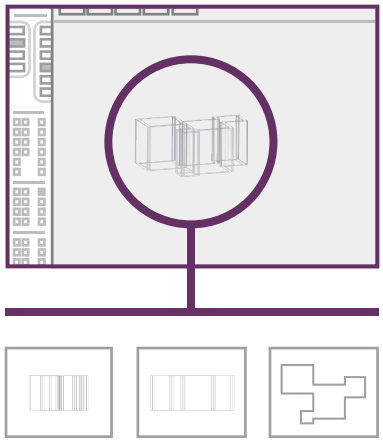
Use of computing applications focuses on creating individual representations of a building, which must be kept in coordination.



NEW: MODEL CREATES OUTPUTS

Use of computing applications aims to collaboratively create and evolve a unified virtual model of a building project.

The information in this unified 3D model can then be used to automatically create all traditional architectural plans.



By comparison, the introduction of CAD had little impact on traditional practice. CAD changed who was doing some things, but the structure of work was mostly the same...

Luckily, everything about this tool seems like it is designed to clarify this new mindset and to help us to build it into the way that we work...

Key *application envisioning* questions:

What overall models could encapsulate the "what and how" of your interactive application's proposed roles in targeted knowledge work? How might those overall "functional stories" be communicated to users? Similarly, how could your team promote clear "sub-stories" for each of your central functionality ideas?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What organizing, big picture mental models do targeted individuals currently have for the work practices that your team is striving to mediate?

How do the mindsets and constraints inherent in different tasks or larger activities drive workers to adopt different frameworks for thinking and acting?

How might your team use your insights into these mental models as a basis for envisioning innovative and compelling conceptual models for your computing tool?

What essential, high level operational approaches in your sketched application ideas could reference and extend upon workers' existing ways of thinking about their efforts?

How might individual functionalities make similar connections to workers' current understandings?

What new and different conceptual foundations might workers need to understand in order to successfully make use of your computing tool in their own practices? How could these new conceptual models be framed by their existing mental models?

How might people in different roles, using application displays that are tailored to their own identities, develop different conceptual models for your sketched computing tools?

How could this impact their common ground for communication?

What might it sound like when a hypothetical user describes one of your proposed conceptual models? What takeaways should they have about your tool's purpose, offerings, and behaviors?

How might your teams' proposed conceptual models be communicated through application design and functional scaffolding? Through other channels, such as informative marketing and introductory instruction?

How could your approaches to communicating target conceptual models tie into your larger ideas about design strategy and brand?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C2. Application Interaction Model

Knowledge work applications can benefit from a consistent and overriding interaction model that defines a computing tool's "shell" of navigation and overall approach to interactivity. Product teams can envision interaction models that are complementary to targeted work practices, appropriate for their sketched design strategies, and framed by workers' experiences with other tools.

Examples from three knowledge work domains:

> A scientist sees that each edge of her analysis application is a panel that can have different impacts on the central visualization area of the tool. One edge controls what data is being displayed, while another controls how selected data is visualized **(see illustration on next page)**.

> A financial trader's new trading application presents four columns, each with a different purpose. The left column has tables that drive what is shown in the next two columns, while the right column shows market data and other trader's action.

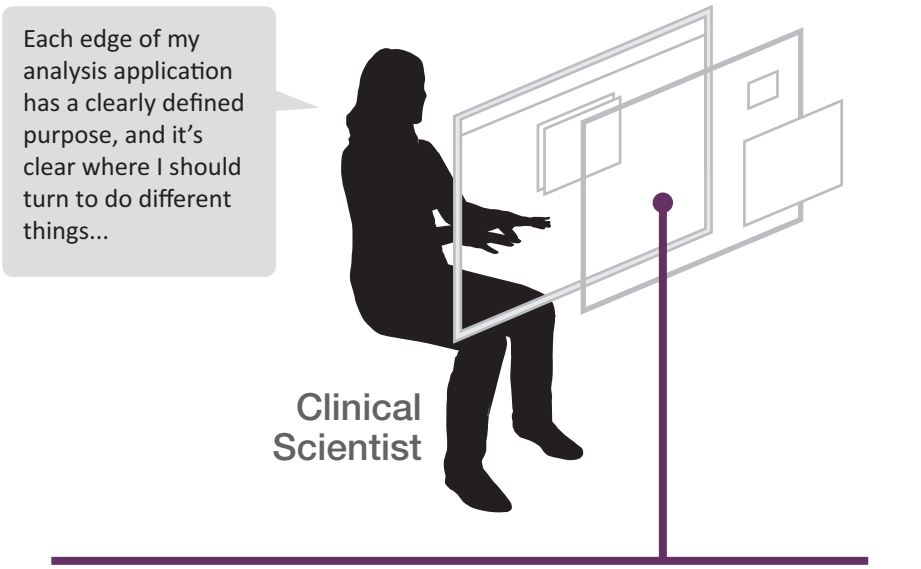
> An architect discovers that her new building modeling application is organized by a series of different views of a project, with each view providing its own set of related functionality. Since the tool seems to have countless functions, she finds this organization method to be very clear and effective.

Interaction models, in the parlance of this book, are the highest level expressions of an application's structure. The population of interaction models used in many knowledge work domains does not contain much useful "biodiversity," and, in general, there is considerable potential for product teams to explore meaningful innovation in this area. Contemporary conventions (L2, C3) are extensively recycled, often with the expectation that these standards will drive efficiencies in product development and adoption.

For many technologists, the selection of an interaction model seems to be simplistically divided between either a general, "menus on the top" workspace model, where tasks are largely open and unsequenced (B4, L1), or a "wizard" like model, where single track processes can be highly constrained (A4, C6). Within these broad categories there is considerable room for tailored solutions, and product teams can improve upon conventional approaches by specifically optimizing them around their sketched concepts for mediating work (A, C5). Concerns about interaction efficiency (D2, D3), multitasking (G5), learnability (D7, K2, K5, K6), findability (C4), and other factors, can drive teams to envision new approaches or consider leveraging specialized interaction models from other domains. Iconoclastic interaction models (L5) can be direct expressions of a product's conceptual models (C1) or novel hybrids of different design patterns.

When product teams do not actively consider divergent approaches for their applications' interaction models, opportunities to appropriately tailor the encompassing structures of computing tools to targeted work practices can be lost. Beyond lost opportunities for targeted innovation, resulting application frameworks may not adequately support the flow of workers' practices or sufficiently communicate a tool's purpose, offerings, and behaviors.

See also: B9, C, L, F, G2, J2, K1, K4, L4



BOTTOM TICKER: Presents collaborator status and time sensitive messaging around the central data-base being visualized

TOP PANEL: Contains all of the controls that determine how data is visualized in the screen's central area

LEFT PANEL: Contains flexible tables that can be transformed to show several different types of relationships in clinical data

RIGHT PANEL: Presents saved snapshots of users' actions, allowing them to retrace and alter their navigation pathways

Key *application envisioning* questions:

What directions can your team generate for the deliberate "shells" of your application concepts, including their approach to containing, enabling, and shaping your sketched functionality ideas? What types of interaction models could effectively support targeted knowledge work in a way that embodies your strategic focus?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What interaction models are commonly found in the computing tools that targeted individuals currently use?

Which models are normally associated with the specific tasks and larger activities that your team is striving to mediate?

What larger design and technology trends could inform your ideation process on this important topic?

Which conventional interaction models could be well suited to the quantity and type of functionality concepts that your team is considering? What benefits could reusing these patterns have for your product's success? What modifications might they require?

What advanced analogies to other types of products might your team draw upon when thinking about possible interaction models for your computing tool?

What types of interaction models could match the embedded conceptual models in your sketched application directions?

Which of your sketched functionality concepts could play a primary role in your interaction model choices? Which should probably not?

How might your team tailor typical interaction model features to better support and encompass certain functionality ideas?

What novel, or even iconoclastic interaction model concepts might you envision? How could these concepts embody valuable approaches for sense making, organization, and interactive flow in targeted work practices?

What could it be like to navigate through tasks and larger activities in the interaction models that your team is considering? Which models could be more likely to provide workers with a sense of compelling engagement and accomplishment?

How could requirements for multitasking, procedural efficiency, or instructional clarity influence your interaction model decisions?

What impacts could different interaction model selections have on brand and branding approaches?

How might your interactive application scale in functionality over time? What impacts could these scaling scenarios have on your team's interaction model choices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C3. Levels of Interaction Patterns

Looking across the sketched functional offerings in a product team's application concepts, there are often opportunities to categorize and standardize certain repeating patterns. Teams can capture and expand upon internal consistencies at different levels of granularity, promoting eventual learnability, usability, and implementation efficiencies within their computing tools.

Examples from three knowledge work domains:

> An architect finds that there is an overall feel of consistency in the design of her new building modeling application, even though each area of the product seems carefully optimized to support different actions. Across the entire application, different tools and dialog boxes are presented in a predictable and clearly mapped manner that makes them easy to interpret and use **(see illustration on next page)**.
>
> A scientist discovers that there are two distinct ways, within the same overall interface, that her new analysis application allows her to act. Common, standard analysis methods are supported by highly directive, step by step screens, while less predictable analyses are supported by a series of flexible workspaces devoted to particular approaches.
>
> A financial trader knows that his trading application has different "categories" of screens. When he navigates to tools and options that he does not use very often, he recognizes smaller components that follow the same mold as screens that he uses repeatedly throughout his day.

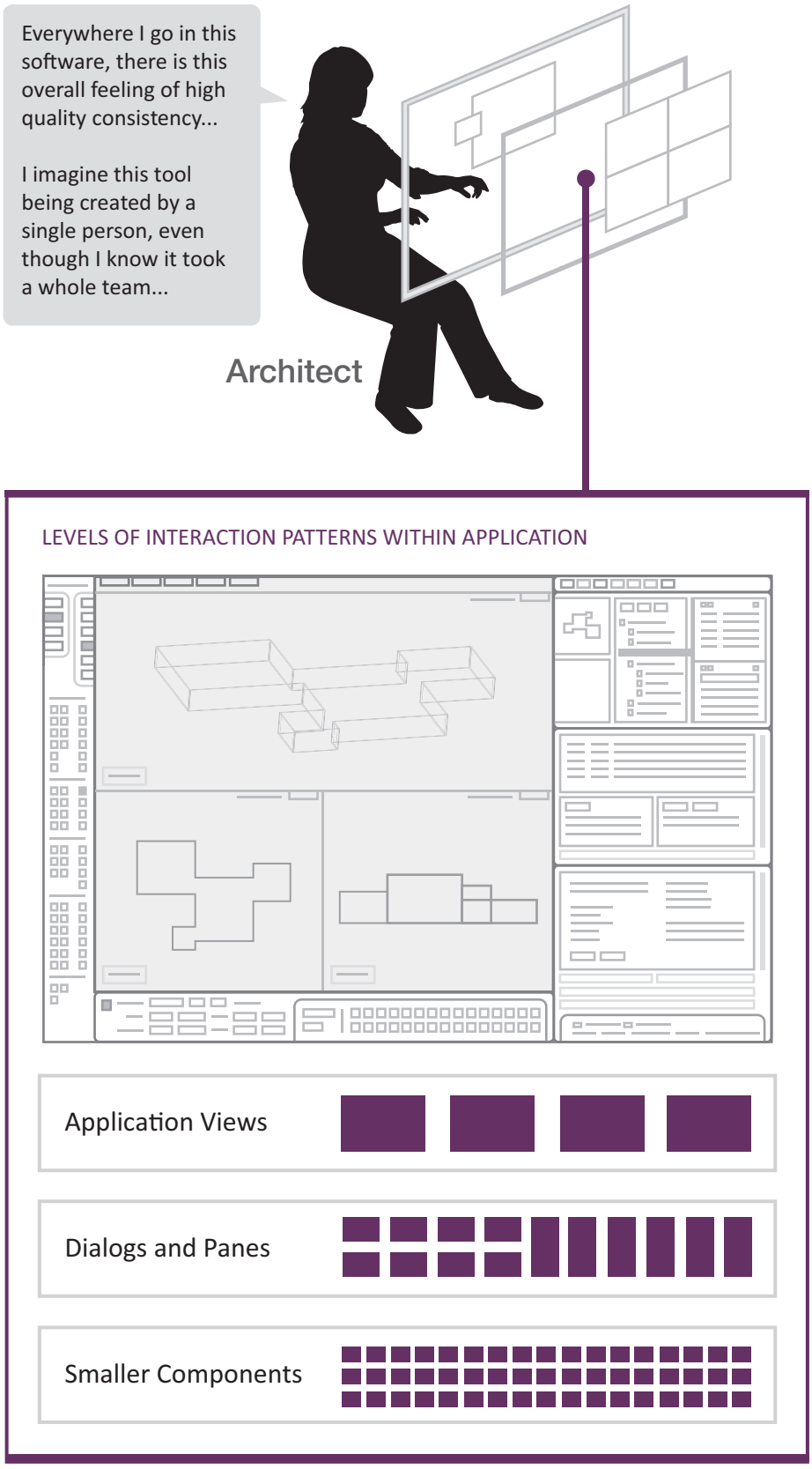
While envisioning applications, many product teams gravitate toward copying low level, "literal" user interface patterns from other products that they, and presumably their targeted knowledge workers, are familiar with. These vernacular design selections are often made on a one by one basis within particular functionality concepts, without considering requirements and consistencies across the entirety of a computing tool (B9, C6).

Product teams can envision more expansive value from interaction patterns by mapping them at multiple levels of convergence, starting at an application's interaction model (C2) and working downward through several tiers of user interface detail (A4, A5). Teams can then experiment with applying their nesting and interrelated patterns across their sketched functionality concepts, envisioning how knowledge workers might transfer their experiences among interactions (C1, D7).

When product teams do not actively consider the potential role of interaction patterns at different levels within their application concepts, resulting inconsistencies may hinder workers' abilities to develop useful expectations. Without these expectations, people may find new or infrequently used functionality more difficult to learn (D2, D3, K2, K5, K6). When inconsistencies are noticeable, they can also negatively impact individuals' perceptions of an application's quality (K12).

Conversely, product teams that focus too heavily on establishing and applying interaction patterns can overlook opportunities to envision design concepts that are highly tailored to the work practices that they are striving to mediate (A).

See also: B1, C, F, G2, G3, J6, L, K1


Everywhere I go in this software, there is this overall feeling of high quality consistency...
I imagine this tool being created by a single person, even though I know it took a whole team...
Architect
LEVELS OF INTERACTION PATTERNS WITHIN APPLICATION
Application Views
Dialogs and Panes
Smaller Components

Key *application envisioning* questions:

Scanning the breadth of your team's promising functionality concepts, what typical or novel interaction patterns might you identify and meaningfully reuse? How might your team organize these valuable regularities into different tiers of patterns within your application proposals, ranging from large to more granular?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What, if any, interaction patterns do targeted individuals expect to see when using computing tools in the work practices that your team is striving to mediate?

What patterns, at different levels of granularity, have become a standard part of how knowledge workers' understand their computing tools?

What value do workers find in these known and expected patterns? What do they think of the conventions that they currently use?

What larger design and technology trends could influence your ideation about interaction patterns?

What advanced analogies to other types of products might your team draw upon when thinking about appropriate patterns?

What inherent consistencies are present within the scope of work practice you are targeting? Based on these consistencies, which of your envisioned functional areas could have strong similarities?

Within the particulars of your sketched functionalities, what smaller consistencies could become internal standards?

How might the reuse of interaction patterns in your application promote the transfer of workers' learning in one interactive experience to other interactive experiences?

Where could the particulars of workers' goals drive meaningful differentiation in interaction design responses, rather than patterned standardization?

When introducing new interaction patterns into workers' practices, what analogies might your team make to known interactivity scenarios?

How might your ideas about your product's larger conceptual and interaction models impact your interaction pattern choices?

How might the emerging language of patterns in your sketched application possibilities relate to, or even establish, the pattern language of a broader family of products in your firm?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C4. Pathways for Task and Activity Based Wayfinding

Effective pathways through interactive applications can be structured to allow knowledge workers to navigate based on the emergent flow of their own efforts. Product teams can derive these pathways from the interrelations between different operations, tasks, and larger activities in targeted work practices.

Examples from three knowledge work domains:

> A financial trader rarely needs to actively locate an entry point to his next action within his trading application. The tool is designed to suit the flow of his work, and he feels like it is "intelligent" enough to "know" the different actions that he might want to accomplish next, providing him with quick ways to get started **(see illustration on next page)**.
>
> A scientist runs a standard transformation algorithm on some of her lab's clinical data. Based on the output of this process, her analysis application presents her with a set of subsequent actions that she may want to perform next in order to further manipulate the content.
>
> An architect copies and drags a new wall into place within her building modeling application. A menu then lists the potential associations and attributes that she could select for the new wall, based on its location relative to other elements.

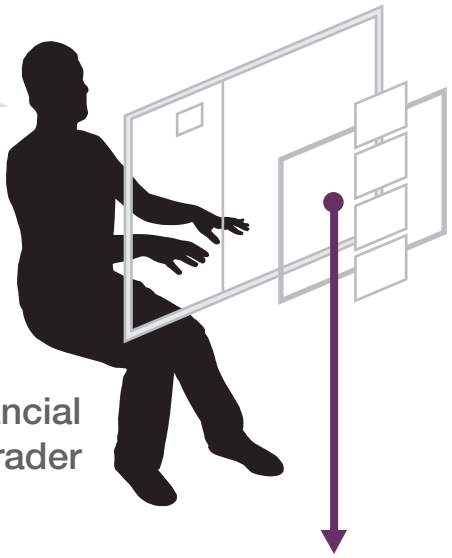
Interactive applications that are tailored to specific knowledge work practices (A) can reflect the flow of those practices back at workers as they accomplish their tasks and larger activities (K3, K13). These reflected flows can allow workers to use their existing mental models and skills to navigate through contextual, progressively disclosed interactions (C3), rather than forcing them to learn how to translate their goals into actions within a tool's own arcane conventions (C1, C2).

Product teams can envision how their proposed mappings between work scenarios and interaction objects (B1, B8) could translate into clear and direct pathways through related interfaces. Teams' early envisioning of these routes can focus on primary work scenarios, at a level of detail that allows them to sketch viable application concepts. Pathway mappings typically become an important part of a product's overall framework (C), communicating a tool's available functionalities and its relevance for targeted work practices (K3). Certain points along pathways can also appear within related functionalities, outside of an application's pervasive "shell," as state based (B5, B6, C10) and contextually relevant navigation options.

When product teams do not actively consider valuable support for practice based wayfinding, resulting applications can feel more like a collection of discrete functions than a cohesive, narrative experience (G1). In entirely disjointed products, workers must first discover what functionality is available to them and then learn to navigate to appropriate functions in sequence. For many knowledge work situations, users may find this sort of wayfinding to be excessively effortful to learn and use effectively (D2, K2, K6) without making errors (C9, G3). These negative effects may lead to a considerable amount of undiscovered or intentionally unused functionality (D3, D4).

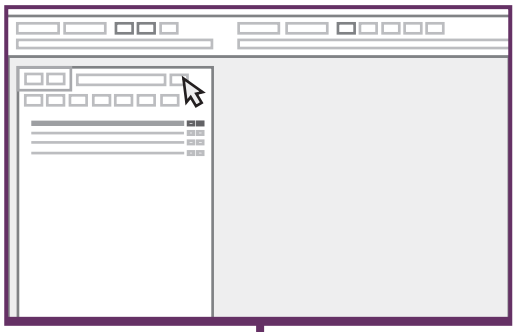
See also: D7, F, G, K5, K8, K9, M1

Even as I make what feel like very different choices, this tool is always somehow stepping me through what I want to do...



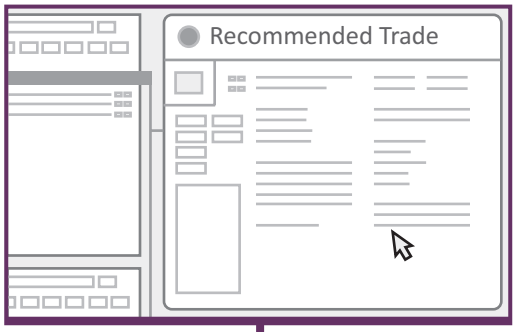
For example, I search for messages from a certain trader at another firm...

And the software highlights the messages from him that it recommends...

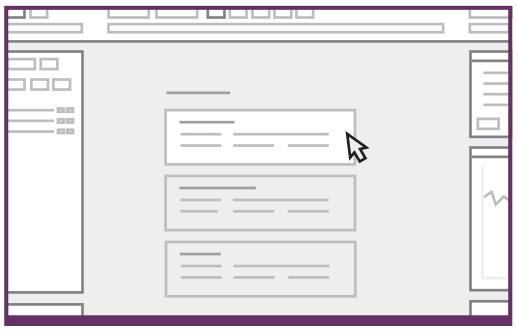


It gives me the option to transform the incoming message into a trade ticket...

And then I go through the highlighted steps to complete the deal...



Next, once that deal is finished, the tool gives me messages right here about what I might want to do next, based on rules that we set up in our group...



Key *application envisioning* questions:

How might your team organize the structuring flow of functional options in your application concepts around understood pathways of meaningful action? How could navigation "naturally" and desirably unfold through the course of targeted knowledge workers' own decisions and efforts within your computing tool?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How might the interrelations between operations, tasks, and larger activities that your team is striving to mediate be reflected in the structural flows of your application concepts?

How might your team situate your sketched functionalities within these essential flows?

What functional areas will contain volumes of content that could benefit from clear, categorical route suggestions?

What might it be like to navigate through different pathways as part of targeted work practices? Which of your team's pathway ideas could be more likely to provide workers with a sense of compelling engagement and accomplishment?

How could interactive routes be made to feel as if they are tailored to the inherent flow of work practice, disclosing content and functionality progressively in order to reduce experienced complexity?

How might the interaction models of your team's application concepts communicate available pathways of action to users? Would workers benefit from a "map" or is it enough to present state based, contextually relevant pathways?

Where could important pathway options be contextually tied into your sketched functionality concepts?

How important is it for workers to have an understanding of where they are in a process? What wayfinding cues could be appropriate in different scenarios?

How directive should interactive pathways be? Where could constrictive, standardizing pathways undesirably limit workers' efforts?

How might the availability of interactive pathways be influenced by application and interaction object states?

What might the experience be like when "turning to" your team's product from work practices that are accomplished outside of the screen, or when transitioning away from your product into other parts of work activity?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C5. Permissions and Views Tailored to Workers' Identities

Application displays that are tailored to knowledge workers' identities can support both organizational goals and workers' own preferred ranges of practice. Product teams can envision how the content and functionalities within their computing tools could be segmented into areas and views that are intended for certain audiences within the same working culture.

Examples from three knowledge work domains:

> A scientist logs into her analysis application to view data in one of her lab's clinical studies, knowing that her login identity will allow her to use all of the tool's many options. She has organized this same study's permissions so that technicians in her lab can only use the analysis tool to upload their completed experiments and manually check the quality of their data **(see illustration on next page)**.
>
> An architect sets up the permissions of a new project in her building modeling application to have different views and options for other architects working on the project; external consultants, such as civil engineers; and even the project's client, who will only be able to view building plans and renderings generated at certain project milestones.
>
> A financial trader knows that, within the trading application that he uses every day, there are data views that are only available to his bosses.

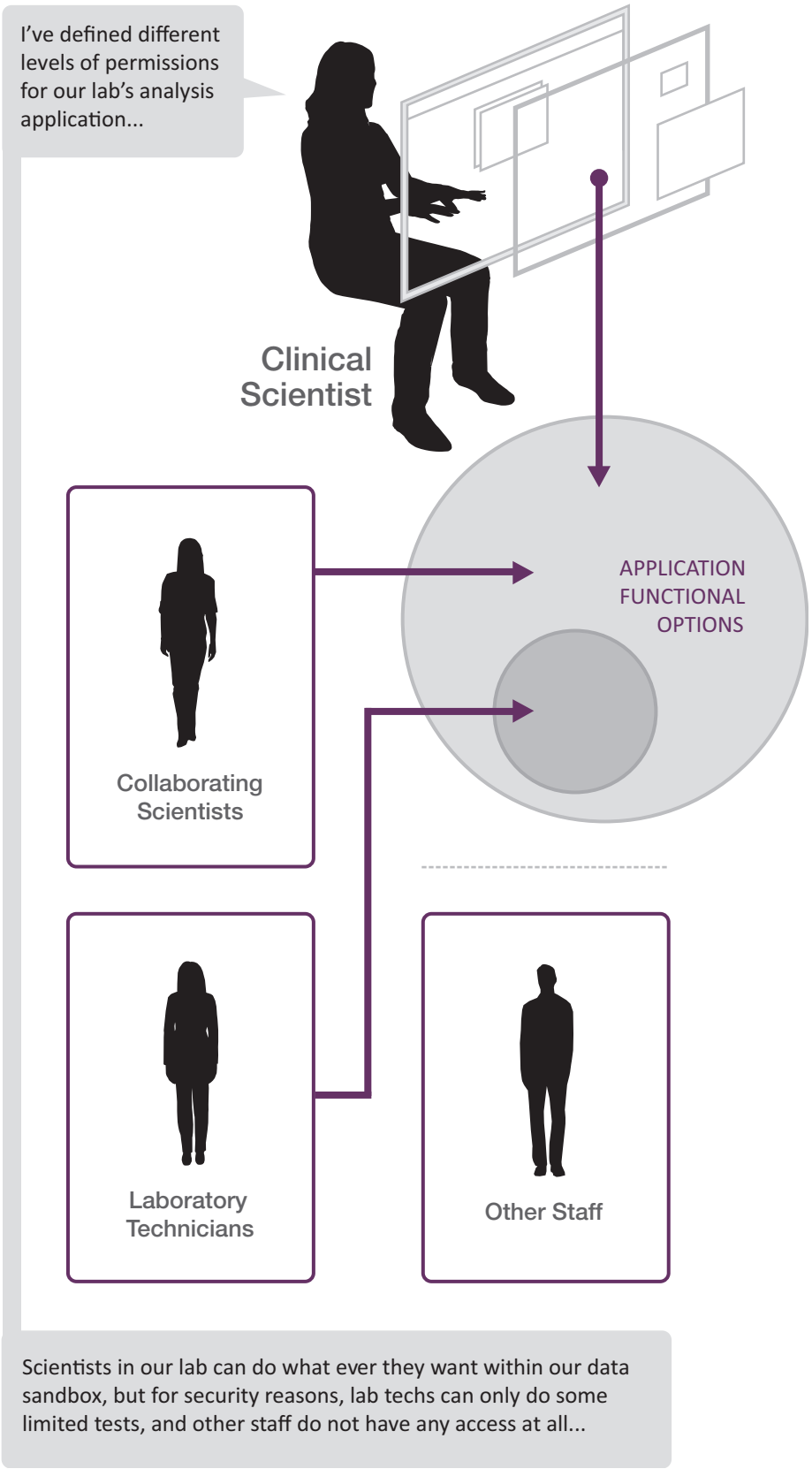
Knowledge workers enacting different roles within an organization may perform different activities (C6) or contribute to the same activity via separate goals and practices (A7, A8). In the context of these established divisions in responsibilities, organizations may have specific requirements for segmenting an interactive application's scope to meaningfully suit categorized identities in their workforces (A1, A2). Permissions features in computing tools can shape each user's ability to see and make use of certain data and interaction options.

Product teams can envision permissions concepts that map to key segmentation needs and desired levels of flexibility. Teams can also sketch structural approaches for tailoring application views to the needs of different user segments, displaying available functionality (A9) and content (B5, B7) in specialized interface layouts and relevant representational forms (F).

When product teams do not adequately consider the potential role of application permissions and views tailored to workers' identities, opportunities to clarify interactions for different audience segments can be lost. Resulting applications may not contain valuable barriers to access that can be essential for supporting specific cultures of work (C8). Some individuals may find that these products present content and functionality intended for "too many different people" at the same time, making these tools excessively effortful to learn and use effectively (D2, D3, K2, K6) without crossing role based boundaries and committing errors (D4, C9, G3).

Conversely, when permissions and tailored views are applied without sufficiently considering potential impacts on collaboration between roles (C7, G4), workers may find it more difficult to establish common ground for communication (F1, J2).

See also: A, B8, C, G7, J, M


I've defined different levels of permissions for our lab's analysis application...
Clinical Scientist
APPLICATION FUNCTIONAL OPTIONS
Collaborating Scientists
Laboratory Technicians
Other Staff
Scientists in our lab can do what ever they want within our data sandbox, but for security reasons, lab techs can only do some limited tests, and other staff do not have any access at all...

Key *application envisioning* questions:

Based on observed role segmentations and security needs in the organizations that your team is targeting, what approaches can you envision for meaningfully categorizing knowledge workers' identities in your application concepts? How might these categories drive differing access and interactions with certain functionalities and content?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What divisions of labor do targeted organizations currently prescribe for the work practices that your team is striving to mediate? What rules currently dictate access to certain workplace artifacts?

How are workers' identities currently managed within targeted organizations? How important, relatively, is the security of these identities?

How flexible or prescriptive are different individuals' roles in observed practice? Do they actually correspond to organizational chart abstractions, or are targeted workers bigger generalists than they often realize or admit?

What larger design and technology trends could influence your team's ideation on the topic of identity management approaches?

How might your team model existing power relationships and levels of responsibility within targeted organizations in relation to your sketched application concepts?

Which functional areas and interaction objects stand out as "belonging" to some categories of specialized workers but not to others?

At what point does it make sense for your team to start thinking of specialized, role based permission sets as fundamentally different views your application, rather than the same interactive frame with some features turned on or off?

How might separate, permissions based views drive desirable simplicity and decreased learning effort?

Where might the splintering of a computing tool into different views introduce undesirable limitations on individual action, opportunities for mode errors, and other breakdowns?

What impact could identity based segmentations have on the larger conceptual and interaction models of your sketched products?

What implications could divergent application views have for collaboration and communication within your computing tool? How will workers maintain common ground?

How might your team's approaches for permissions and identity based views relate to your functionality concepts supporting cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C6. Standardized Application Workflows

Some cooperative processes in knowledge work can be supported by computing functionalities that facilitate entire sequences of standardized effort. Product teams can envision functionality concepts that could valuably distribute segments of larger work processes among multiple users; however, restrictive workflows may not always be an appropriate design response.

Examples from three knowledge work domains:

> A financial trader wants to execute a trade that is so large that it requires signoff from his manager. The trading application displays a large notification on the manager's screen, and the two coworkers shout back and forth across the trading floor about the merits of the potential deal. Eventually, the manager indicates his approval in his own view of the trading application, which then executes the pending transaction **(see illustration on next page)**.

> A scientist sets up a work request in her lab's information management application so that a certain technician will rerun some clinical samples. When the technician receives this work request, he can quickly translate the experimental protocol into his own laboratory processes, run the samples, and then upload the new data for the scientist to review.

> An architect defines a standardized workflow in her building modeling application that will usefully drive how her team collaboratively uploads, evaluates, and categorizes early ideas for a new building project.

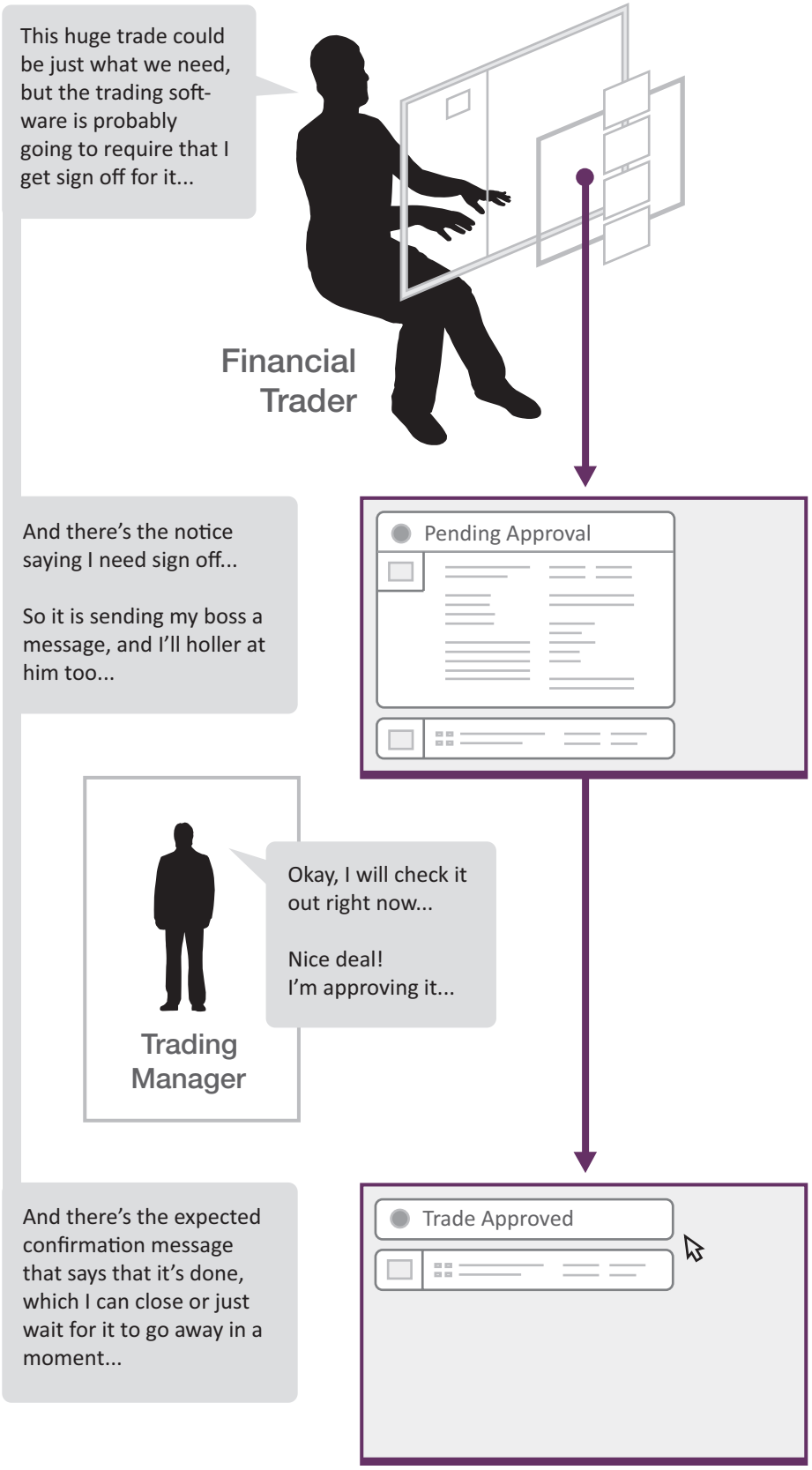
Established practices in knowledge work professions may bear little resemblance, either literally or in spirit, to highly standardized, "scientifically managed" assembly lines. It can be important to recognize, however, that even within otherwise variable activities (A7, A8) there may exist some consistent, sequential segments of established and repeated work process (A4). Requirements for these workflow standardizations can arise from individual workers, their organizations, or larger communities of practice.

Product teams' concepts for mediating workflow processes can have substantial impacts on their sketched ideas for cross functional application frameworks (C1, C2). Computer mediated workflows can outline the number and type of steps to be completed, levels of instructiveness (K2, K7), how handoffs will occur (C5, G7, J3), and many other important factors. Knowledge workers may value how these process oriented functionalities reduce undesired effort through automation (E3, E4) and distribution of efforts to colleagues (J). Appropriate workflow tools may also improve the predictability and quality of cooperative outputs (C9, G3, L1).

When product teams do not actively consider the potential role of standardized workflows in their application concepts, opportunities to translate existing workflows, or to create new value in workers' experiences of larger processes, can be lost.

Conversely, highly skilled knowledge workers may not always value novel standardization that is rooted in distant notions of efficiency, such as those sometimes outlined in the name of "business process redesign" (D1, G5).

See also: A, B3, B5, C, D, E, F1, K1, K3, M


This huge trade could be just what we need, but the trading software is probably going to require that I get sign off for it...
Financial Trader
Pending Approval
And there's the notice saying I need sign off...
So it is sending my boss a message, and I'll holler at him too...
Okay, I will check it out right now...
Nice deal! I'm approving it...
Trading Manager
Trade Approved
And there's the expected confirmation message that says that it's done, which I can close or just wait for it to go away in a moment...

Key *application envisioning* questions:

What portions of the knowledge work that your team is targeting truly follow standardized and routine processes — but still require human judgment and action? How might your application concepts meaningfully structure and usefully reduce burdens in these procedural flows for all involved?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which tasks or larger activities that your team is striving to mediate are currently part of standardized workflow processes?

What other artifacts and technologies are involved in these processes?

What value do current workflows provide in targeted organizations?

What are the individual measures of success for different segments of existing workflows? For entire workflows?

What work processes do both knowledge workers and their organizations want to standardize further? Where might organizational goals for workflow crystallization show a clear mismatch with workers' goals and preferred ways of practicing?

How could your application concepts maintain or expand upon the value of existing workflow processes? How might they provide valuable new workflow options?

When might it be more appropriate to support structured work with integrated communication channels and clear object ownership rules, rather than regimented and inflexible workflow tools?

How much visibility might workers want into their colleagues' activities and workflow contributions? What value could this visibility provide?

How might separate workflow views for different contributors drive desirable simplicity and decreased learning effort?

How might your team's sketched workflow functionalities support interpersonal interaction and communication at key junctures?

What role could flexibility and customization play in your workflow concepts? At what point might desirable variabilities challenge the usefulness and viability of standardized workflow functionality?

What impact could extensive workflow offerings have on the larger conceptual and interaction models of your sketched products?

How might your team's approaches for standardized workflows relate to your functionality concepts supporting permissions, identity tailored views, cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C7. Structural Support of Workspace Awareness

Valuable functional support for cooperative or collaborative knowledge work activities may impact the larger structure of a computing tool. Product teams can envision pervasive cues within their application concepts that could highlight significant actions of other users acting in the same "workspace."

Examples from three knowledge work domains:

> An architect knows that one pane in her building modeling application contains a variety of information about what her collaborators are doing in the same project file. She uses this pane to understand who is working in the same building areas that she is, as well as to see who is available for conversation **(see illustration on next page)**.
>
> A financial trader learns that a certain area of his trading application will visually indicate when another trader is acting on the same information that he is. This notification allows traders to prevent discoordination by initiating discussions about their current work.
>
> A scientist knows that any time she looks at individual items in her analysis application, such as samples within a clinical study, a specific area of the screen will notify her of related actions being performed by other members of her lab. Also, when she first logs into the application, a special "welcome" display summarizes key changes that have been made to lab data since she the last time she accessed the tool.

Knowledge workers are often highly skilled at understanding how their own actions fit into the context of cooperative and collaborative activities in their organizations. Computers can have dramatic impacts on this understanding. For example, when interactive applications become a key focus of work practice, implicit visibility and communication (J1) that was once tied to the physical performance of work can easily become hidden or entirely lost (G4).

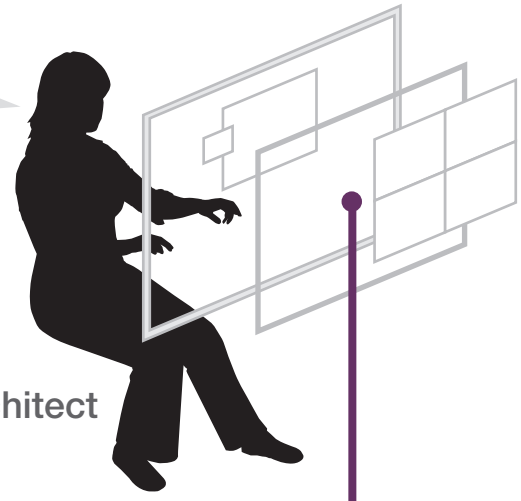
Product teams can envision structural cues that could promote useful types of workspace awareness across the range of tasks and larger activities that they are striving to support (A7, C). An application's larger framework can include functionality for contact facilitation (J4) and other features that highlight shared opportunities or potential conflicts within a networked environment (B7, J5, H3). These structural responses can dramatically impact a product's conceptual models (C1), interaction model (C2), and permissions functionalities (C5).

When product teams do not actively consider how workspace awareness could be incorporated into an application concept's cross functional framework, opportunities to promote cooperation and collaboration can be lost. Even though envisioning workspace awareness on a function by function basis can solve individual collaboration issues (G4), without structural support, collaborators may find that they have difficulty planning larger activities (D2, D3), communicating effectively (J), distributing effort (E), and preventing conflicts (C9, G3).

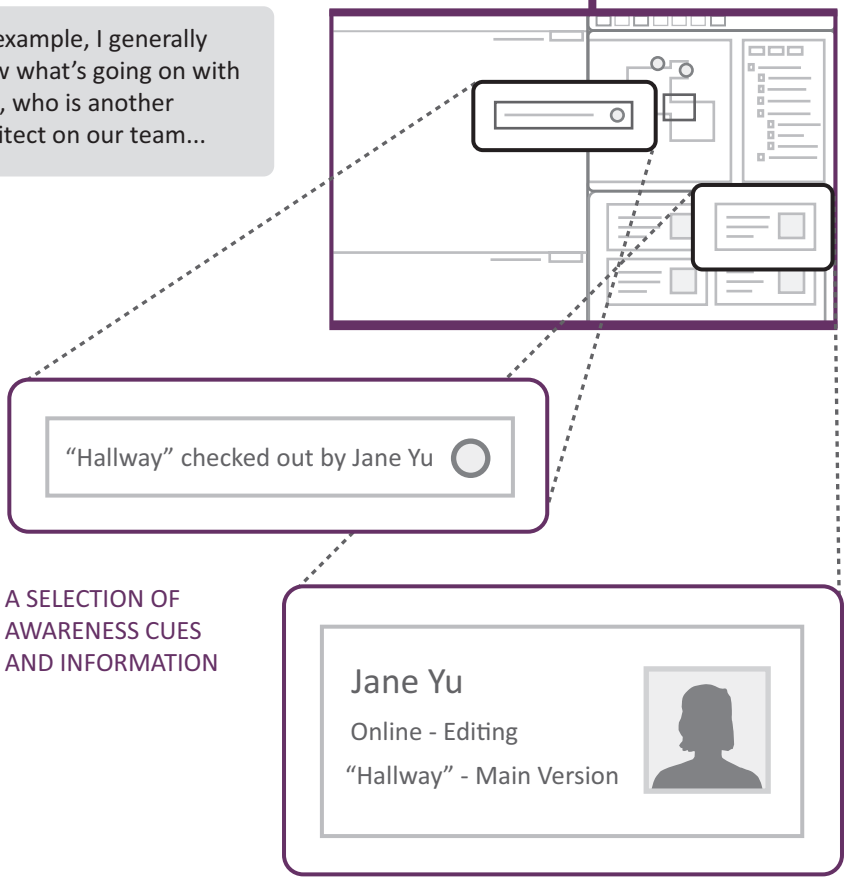
Conversely, too much visibility into the actions of others can be distracting (D4) and can potentially lead to unwanted surveillance effects (A2, G7).

See also: A, B, F1, G, H, J2, J3, K, M1

My building modeling application always lets me see at a glance what my colleagues are working on, without going out of my way to look...



For example, I generally know what's going on with Jane, who is another architect on our team...



The accumulation of these little clues really changes the amount we have to communicate, as well as the topics that our team talks about when we do chat face to face...

Key *application envisioning* questions:

What structural, application level approaches might your team envision to allow targeted knowledge workers to stay usefully and meaningfully aware of others' actions within the same data locale? What might these awarenesses feel like in practice?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently keep track of their colleagues' actions as part of the work practices that your team is striving to mediate?

How, specifically, do current forms of shared awareness promote the effective execution of loosely coordinated or truly collaborative work? How do they prevent conflicts?

What breakdowns currently occur due to insufficient awareness? Could these problems present opportunities for your product?

Where might the introduction of your team's computing tool remove implicit and subtle awareness cues from targeted work practices?

What larger design and technology trends could influence your ideas about how workers might remain appropriately aware of others' actions within your application concepts?

How might your sketched application frameworks aid workers by providing valuable workspace awareness cues at a structural level, across various functional areas?

What types of actions in your product's shared workspaces could be tied to stronger, attention grabbing cues? To weaker, almost subliminal, cues?

Who needs to see various cues? How might awareness information relate to individuals' permissions and tailored views?

At what point might users of your computing tool face information overload from awareness cues? When might it be more appropriate to tie workspace awareness to individual functions and contextual areas, rather than your tool's overarching structure?

How long should specific awareness cues last in your application's framework? How might they be tied to longer term, stored histories for certain functions and interaction objects?

What impact could overarching awareness functionality have on the larger conceptual and interaction models of your sketched products?

What unwanted surveillance effects could unintentionally occur from broadcasting specific actions to other workers?

How might any standards set by your structural workspace awareness designs influence your team's envisioning of awareness cues within individual functionality concepts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C8. Defaults, Customization, and Automated Tailoring

Knowledge workers may want to make persistent changes to default settings in order to tailor how they interact with a computing tool. Product teams can endeavor to create useful defaults; provide clear, consistent, and direct means of changing them; and consider scenarios for useful automation around some setting changes.

Examples from three knowledge work domains:

> A scientist modifies a certain parameter to influence how her analysis application will automatically compute a derived clinical variable. After double checking the effects of this parameter change within her most commonly used visualizations and procedures, she sets the modified value as the default setting for all new studies **(see illustration on next page)**.
>
> An architect finds that the input settings of a drawing tool in her building modeling application are making some parts of her work unnecessarily painstaking. She navigates to a single screen that contains all of her application preference settings and decreases the particular tool's sensitivity to input.
>
> A financial trader updates important automation defaults in his trading application that dictate how the computing tool will adaptively fill in proposed information under different circumstances.

In specialized products for knowledge work, a single parameter can make or break the effectiveness of an entire system. Product teams can envision default settings for their interactive applications that are optimized to cover the most common scenarios of use (A4) or the broadest variety of work practice (A6, A7, A8).

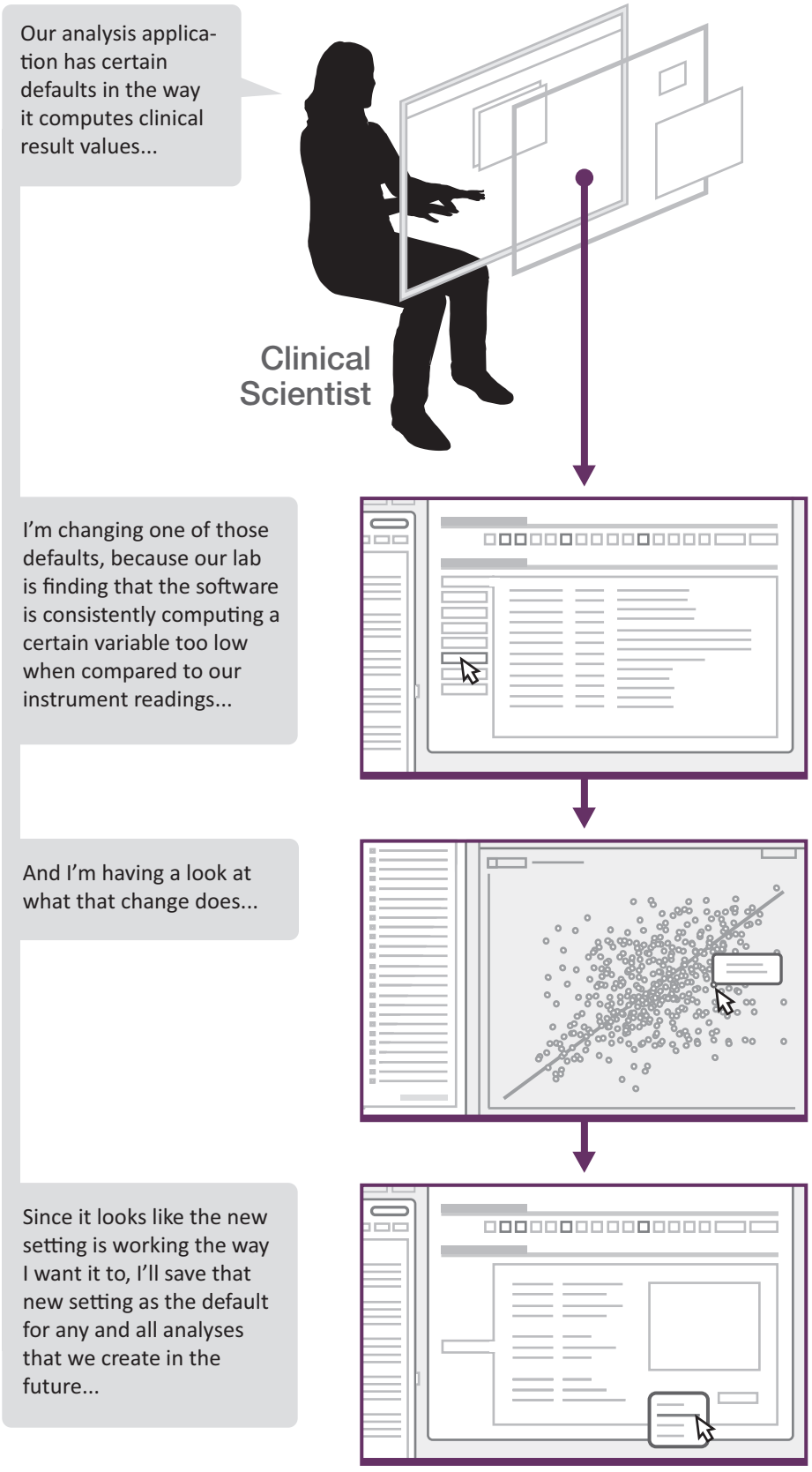
When default settings have the potential to shape workers' interactions or outcomes in ways that are not in alignment with their goals, applications can provide customization functionality that allows for local modification of key parameters (C5, F8). Product teams can envision these customizations at the level of individual workers, larger groups, or entire organizations (B10).

In carefully selected cases, workers may appreciate suggested or automated tailoring of settings (E3) based on their logged behaviors within an application. To avoid confusion, definers and designers can envision ways to clearly communicate these adaptive changes (B6, F11, H4) as well as provide methods to easily reinstate earlier values (E6).

When product teams do not sufficiently consider the potential role of defaults, customization, and automated tailoring, resulting computing tools may not be suitably configured or configurable for the particulars of knowledge work. Opportunities for close alignment with work practices can be lost, and individuals may struggle through their adoption experiences (K), potentially creating and enacting excessively effortful work arounds (D2, D3).

Conversely, extensive changes to defaults may reduce representational common ground between workers (F1, J2) that is often needed for effective communication (J) and collaboration (B7, C7, G4).

See also: A, C, F, I, K11, M4


Our analysis application has certain defaults in the way it computes clinical result values...
Clinical Scientist
I'm changing one of those defaults, because our lab is finding that the software is consistently computing a certain variable too low when compared to our instrument readings...
And I'm having a look at what that change does...
Since it looks like the new setting is working the way I want it to, I'll save that new setting as the default for any and all analyses that we create in the future...

Key *application envisioning* questions:

How might your team clarify and reduce the effort needed to understand and set important parameters in your application concepts? How could the interplay of appropriate default values, manual customization, and automated tailoring enhance your product's effectiveness across a breadth of targeted contexts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which variabilities in the operations, tasks, or larger activities that your team is striving to mediate might lead to a genuine need for customization options?

Which default settings in your team's application concepts will individuals and organizations expect to have some control over? Why?

Which settings stand out as pivotal in your team's sketched ideas for work mediation? Which will probably not capture workers' interests and may only be accessed rarely, if at all?

What larger design and technology trends could influence your ideas about defaults and local tailoring of settings within your computing tool?

Which defaults in your application concepts could be optimized by covering the most common scenarios of use in targeted organizations?

Which defaults might be better optimized by considering the broadest variety of work practice?

Which parameters might be impossible for your team to set defaults for without local input from individual workers or their organizations?

Where could automated tailoring of settings be appropriate, useful, and clearly executed? Might it be more appropriate for such automations to suggest changes that workers could then select as customizations?

How might the scope of a single setting change apply to individual workers, larger groups, or entire organizations?

How could a central area for settings changes within your application's framework enhance the clarity of related tasks?

How might new or unexpected changes to defaults be flagged and meaningfully communicated?

What negative impacts could changes to defaults have on cooperative and collaborative work? How might these impacts be mitigated?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C9. Error Prevention and Handling Conventions

To ensure that potential errors in mediated knowledge work are preempted and managed in a consistent and appropriate manner, product teams can develop internal conventions for their application concepts. These standards can promote learnability, usability, and implementation efficiencies.

Examples from three knowledge work domains:

> A financial trader notices and appreciates the different ways that his trading application prevents him from making errors during the course of his typically hectic day. For example, the tool presents small, informative flags as he enters problematic data so that he can make corrections in real time. And, when he tries to complete an action that does not match established business rules, the tool overlays clear, cautionary messages with suggestions for action **(see illustration on next page)**.
>
> An architect has learned that her building modeling application provides constraints on her actions that prevent her from making errors in a categorical variety of ways, whether she is "sculpting" the shapes of a building design or entering data about the properties of a particular onscreen object.
>
> A scientist wants clear, highly visible messages in her analysis application that prevent her from making predictable and common data entry mistakes while she creates new studies. However, for any tasks that involve exploring her lab's clinical data, she only wants messaging of possible errors, without any hard limitations on her actions.

Within the complex mental operations and symbolic abstraction of computer mediated work practices, we can safely assume that people will make errors (G3). The best envisioning response to a recognized possibility for user error is often to design away the conditions under which it might arise.

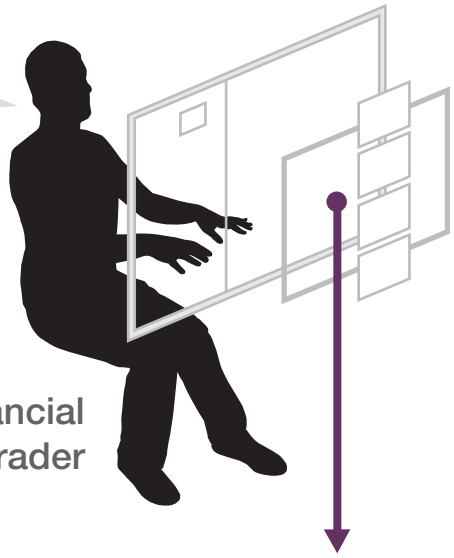
In cases where there are conflicting requirements and high flexibility needs, product teams may find it difficult to prevent errors strictly by envisioning behavioral constraints in their functionality concepts. Teams can meaningfully categorize these stubborn error cases, based on their severities and interaction consistencies (A4). They can then envision patterns for error prevention and handling to apply throughout their sketched application concepts (C3, E3, F10).

Teams can choose many of these patterns from among the error conventions that are commonly used in contemporary interface design (D7, L2). Some products may contain unusual, domain specific error classes (A) that could benefit from ideation of novel, tailored patterns or display formats (F).

When product teams do not actively consider potential conventions for preventing and handling errors, resulting inconsistencies (K13) may frustratingly hinder a worker's ability to effectively accomplish important goals (L1). People may have more difficultly learning such applications (D4, K5, K6, K7) and recovering from mistakes made while using them. Additionally, their perceptions of product quality and utility may decline (K12) as they create and enact defensive work arounds (D2, D3), such as active versioning of valued content (H1).

See also: B6, C, E6, H2, H3, J4, J5, K5, M

Traders have fat fingers like everyone else using a computer, and this trading software steps in to help prevent all sorts of problems in a predictable way...



Like if I'm typing a price wrong, this tool doesn't let me get too far before telling me about it...

Which looks similar to the very useful message that comes up if I'm entering a quantity for a security that exceeds our holdings...

Which is similar to the error stopper that appears when a trade ticket's contents happen to go against the complex mesh of no-trade rules that our group is always updating...

Key *application envisioning* questions:

Looking across the functionality concepts in your team's sketched application possibilities, what common classes of error situations might you identify? What interaction patterns could consistently and appropriately prevent or handle each of these error classes?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What error scenarios are targeted individuals currently concerned with in the operations, tasks, and larger activities that your team is striving to mediate? Why?

How do they currently prevent and handle these errors? Could these situations present opportunities for your product?

Where might the introduction of your team's computing tool create new possibilities for human error in targeted work practices?

How might you divide up the pool of potential error scenarios that you have identified into meaningful classes? How could different approaches and perspectives on this categorization provide insights?

What larger design and technology trends could influence your ideas about preventing and handling classes of errors within your computing tool?

What existing conventions, from a broad selection of interaction patterns in contemporary computing, are most relevant to the error classes and categories that your team has identified?

What domain specific error scenarios might present opportunities for your team to envision useful and specialized error prevention or handling conventions?

How could different levels of error severity be clearly and consistently communicated throughout your application concepts?

What special standards might your team envision to prevent critical errors in highly directive ways? On the other side of the spectrum, what classes of errors do not require such strict prevention and should leave users in the locus of control?

How might these internally consistent standards become a complementary element of your product's larger aesthetic direction and brand? What tone and appearance could be most appropriate for these textual and visual languages?

How might your sketched error management standards relate to, or even establish, the pattern language of a broader family of products in your firm?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

C10. Predictable Application States

High level state information can allow knowledge workers to assess whether an application is functioning properly, decide what avenues of action are currently available to them, and plan the ongoing flow of their efforts. Product teams can envision clearly defined, appropriately simple, and well communicated overall states for their computing tools.

Examples from three knowledge work domains:

> A scientist knows that some options in her analysis application are unavailable during intensive automated processes, such as importing large clinical data sets or running extensive analyses. Since the actions that are currently available during different states are always obvious, it is easy for her to figure out what work she can accomplish while the product is processing complex requests **(see illustration on next page)**.
>
> An architect indirectly learns all of the states of her building modeling application during the course a single project. She now knows that the tool behaves in special ways when it is, for example, opening a building model, creating a detailed rendering, displaying problematic areas of a design, or as occasionally happens, experiencing technical difficulties.
>
> A financial trader expects his trading application to run at top speed whenever he turns to use it. If an issue does arise within the product's operations, he wants the tool to be "smart" enough to detect the problem as soon as possible and then tell his team what to do about it.

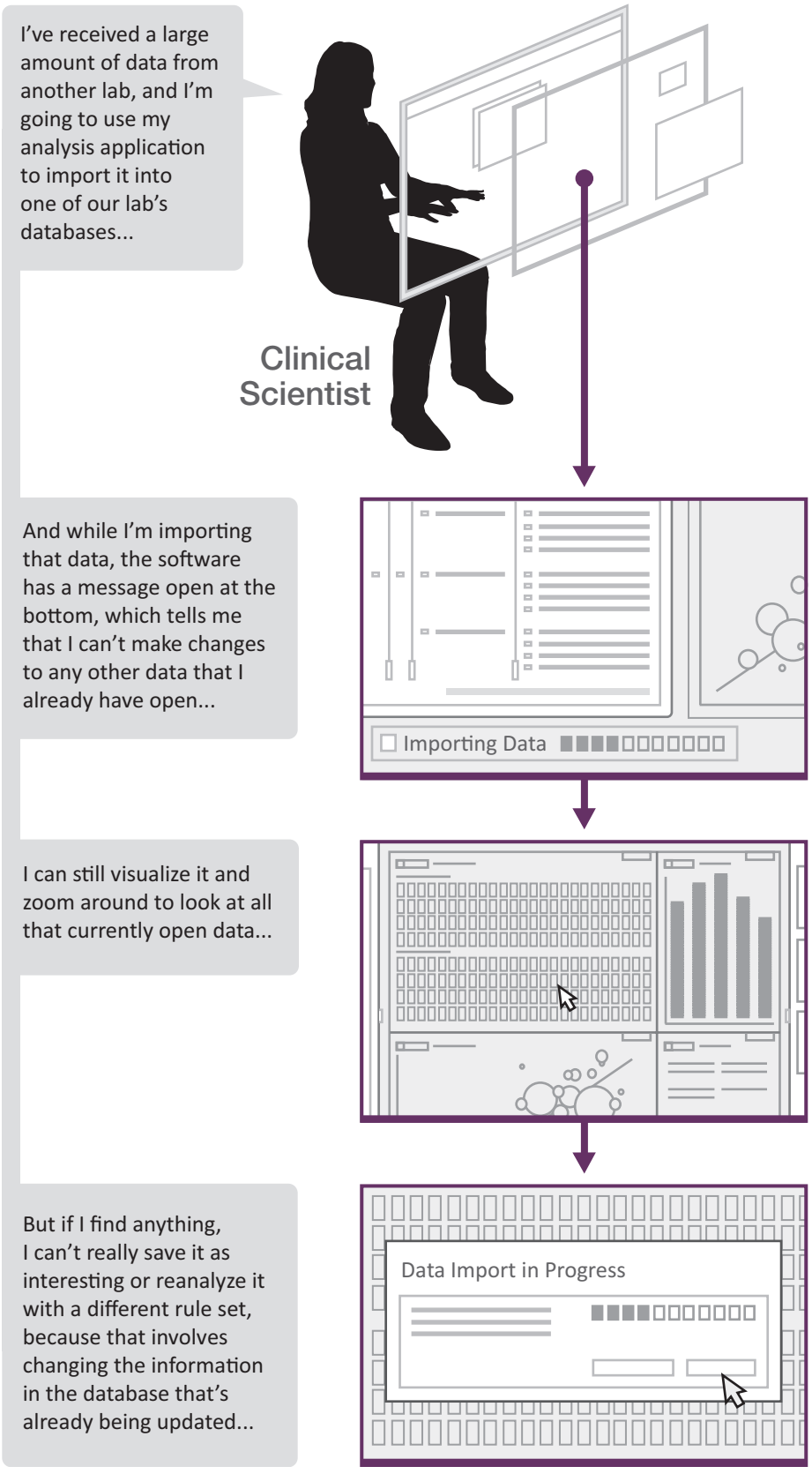
With limited processing resources, network constraints, and other technical bottlenecks, many computer mediated processes in knowledge work are inevitably experienced as something slower than real time responsiveness (E3, E4). For example, users of an application may quickly learn that their valued tool needs to extensively initialize when it is launched and take time to save settings (C8) when work is concluded.

Deliberate, controlled pauses in interaction can also be implemented by design. At certain times, actions may be disabled within an application (C4) as a means of preventing errors (C9, G3) or directing workers toward certain responses.

Product teams can envision appropriate states for their application concepts with an intentional focus on clarity and simplification. Workers do not typically need to be aware of many of the internal, "behind the scenes" conditions of their computing tools (K10, K11). Instead, teams can focus on identifying application states that could directly influence the flow of work activity (D4), such as conditions tied to crucial error cases or lengthy processes where indication of progress could be useful (E5, F6, K4).

When product teams do not actively consider how to define and effectively communicate application states, resulting products may cause confusion as workers attempt to translate their goals into situated actions or longer term plans (D3). When computing tools present vague or confusing states, users may have difficulty developing useful expectations (D2, K2, K6, K7) as well as accurate conceptual models of how a tool essentially works (C1).

See also: A, B5, C, D6, D7, F10, H2


I've received a large amount of data from another lab, and I'm going to use my analysis application to import it into one of our lab's databases...
Clinical Scientist
And while I'm importing that data, the software has a message open at the bottom, which tells me that I can't make changes to any other data that I already have open...
Importing Data
I can still visualize it and zoom around to look at all that currently open data...
But if I find anything, I can't really save it as interesting or reanalyze it with a different rule set, because that involves changing the information in the database that's already being updated...
Data Import in Progress

Key *application envisioning* questions:

What useful or necessary overall states might your team envision for your application concepts (e.g. starting, loading, normal, critical error)? How might these states consistently communicate how your tool is currently operating, what it can currently be used to accomplish, and when, if applicable, its state will likely change again?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What technical limitations might your team's computing tool face when it is run on the available infrastructures of targeted organizations?

How might integration into other systems, or use of networked resources, affect goals of near real time responsiveness?

What interaction cases, looking across the breadth of your sketched functionality concepts, might benefit from an application state that could appropriately direct users' actions?

Which interactions or automated processes in your sketched functionality concepts are likely to require processing time that workers will probably experience as a period of waiting?

What critical errors could become default application states by blocking action until they are resolved?

What set of high level states could comprehensively cover key scenarios in your team's application concepts?

How much detail is too much detail when considering your list of application states? When might several states that are namable and very different from your team's own perspective be better presented to your product's users as a single state category?

How might simplicity in application states promote more accurate conceptual models of your computing tool?

What interactive pathways, due to technical requirements or defined constraints in work processes, could need to be disabled during certain states?

How might application state information be communicated both in your product's overall framework and as part of certain functionality concepts?

Where might these state categories become too confining for variable work practices? How might constraining states be eliminated through redesign of your sketched interaction models?

Which state categories could be more appropriately envisioned at the level of interaction objects rather than presiding at the application level?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D.

Considering Workers' Attentions

Valued computing tools can desirably "fit" into the flow of thinking work: easing burdens, removing distractions, and allowing people to focus on challenging problems.

Designing for such a compelling pairing requires a careful examination of current and potential demands on peoples' attention.

During *application envisioning*, product teams can evaluate and explore how their sketched offerings might impact the allocation and sequence of knowledge workers' efforts.

By taking time to explore the topic of attention related needs and goals, teams can highlight opportunities to tailor and extend their products in truly useful and humane ways.

In many professions, knowledge workers' attentions can be stretched to their proverbial limits by multiple threads of activity and steady streams of interruptions. To be successful in their chosen vocations, workers may become skilled at estimating the effort that incoming work items will require, attending to pressing items, setting aside less urgent needs, and recognizing opportunities to delegate work or otherwise make it less time consuming.

The overall "load" placed upon workers' emotions and cognitive abilities can be a serious consideration for the design of interactive applications. The stress involved in some knowledge work professions has been tied to health problems in the people that practice them. While poorly designed onscreen applications can be obvious contributors to this stress, even carefully designed products, in conjunction with related workplace demands, can require taxing levels of concentration that may be difficult to effectively sustain.

Although many product teams may briefly discuss the limitations of hypothetical users' attentions when they evaluate detailed design options, they may be less likely to do so when considering their products' overarching design strategies and possible design directions. Without these early discussions about potential influences on knowledge workers' concentrations and mental efforts, teams may not recognize certain opportunities to thoughtfully shape their design concepts around important attentional considerations — until after workers have actually adopted resulting products and begin asking for certain types of improvements.

This category contains 7 of the 100 *application envisioning* ideas in this book:

D1. Respected tempos of work

D2. Expected effort

D3. Current workload, priority of work, and opportunity costs

D4. Minimizing distraction and fostering concentration

D5. Resuming work

D6. Alerting and reminding cues

D7. Eventual habit and automaticity

Product teams can use these ideas to explore potential roles for their computing tools in the ongoing flow of knowledge workers' attentions. Ideation around workers' attentional demands in specific situations can help teams establish key design constraints and drive exploration of appropriate design responses. It can also allow teams to uncover opportunities to reduce attention related burdens through targeted automation functionalities.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Providing opportunities to offload effort" (E), "Clarifying central interactions" (G), and "Promoting integration into work practice" (K) categories.

D1. Respected Tempos of Work

Knowledge work can have implicit paces and timings, based in part on workers' inherent mental and physical limitations as human beings. By exploring potential changes to the pacing of individual tasks and extended activities, product teams can meaningfully envision how their interactive applications might impact important tempos in workers' practices.

Examples from three knowledge work domains:

> A financial trader follows a very similar schedule on most every working day. During large parts of this daily routine, he has the potential to be overwhelmed with a steady stream of incoming, discrete decision tasks, most of which are facilitated by his high performance computing tools **(see illustration on next page)**.
>
> An architect's projects typically span over months or, more commonly, years. Her work days are often long, with a sustained intensity level that often leads her to feel hurried as she switches between very different tasks during different project phases.
>
> A scientist is under pressure to quickly understand clinical data in order to deliver exacting discoveries— a "quickness" that she feels on very different time scales. She appropriates whatever computing tools she can to clarify the big pictures of her lab's experimental outputs, while at the same time dealing with a myriad of time sensitive details that are needed to keep her studies running effectively.

Recurring tempos in knowledge work can arise from a variety of factors to become an essential aspect of workers' experiences (A, G1). Expectations for tempos can be set by professional standards, by specific organizations or communities of practice (A7, A8), and by individual knowledge workers, who may establish rhythms to bring their efforts into both internal and external equilibriums.

Product teams can model how established tempos in knowledge work nest into one another, run in parallel threads, or interrupt each other (A5). They can identify tempos in specific practices, tempos in daily cycles, unique tempos for individual roles, and collective tempos across the course of longer term, shared goals within an organization.

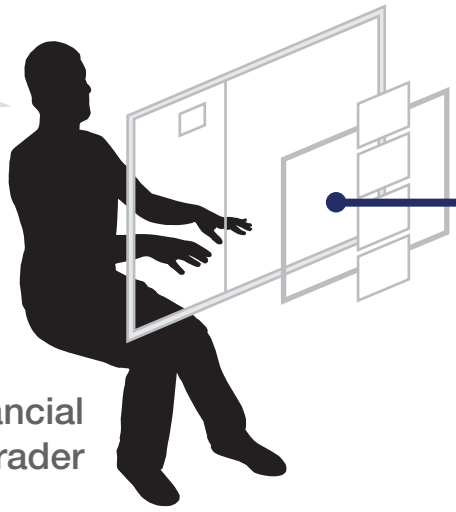
Interactive applications can have major impacts on existing tempos, to both positive and negative effect. As workers adopt a computing tool (K), they may compare the rhythms implied by its pathways to their own expectations and preferences. Valued automation of time consuming and tedious work (E3, E4) can contribute to a positive evaluation. Products that force unwanted changes in tempos without supporting a worker's internal locus of control (E6) may contribute to a negative impression, as well as a sustained elevation in stress level.

When product teams do not actively consider how their application concepts could influence existing tempos in knowledge work practices, they run the risk of creating tools that are out of step with users' desires and needs. Resulting applications may "push" workers through processes too quickly (C6), or perhaps more commonly, enforce interaction pathways that are too slow and extended relative to conventional or desired pacing (C4, D3, D4).

See also: D, C8, E5, J1, J3, K6, K1, K13, M1, M4

My work has a definite rhythm to it that actually helps me to think more clearly...

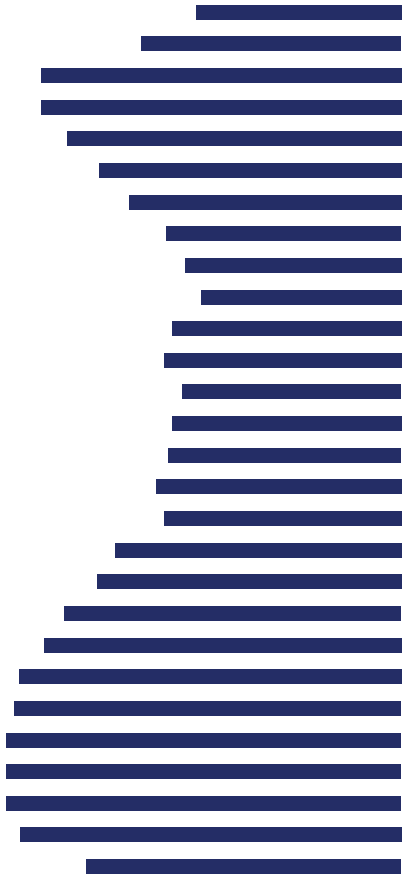
During the really busy times, I won't use any part of my software that slows me down...



Financial Trader

MORNING

In the morning, trading volume can be high as everyone comes in and trades on new information from after the closing bell of the previous day...



Toward the end of the day, when the market is moving fast before closing time, I really need my tools to respond rapidly and to understand what I want to do...

END OF DAY

Key *application envisioning* questions:

How could the interactive flow of your team's application concepts desirably reflect the inherent pacing of targeted knowledge work practices, rather than force unwanted slowing or acceleration in users' experiences? Where might positive shifts be possible?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What tempos are currently found within the tasks and larger activities that your team is striving to mediate?

How did these tempos originate? What factors have perpetuated them?

How can certain paces and timings in different threads of knowledge work nest and inter-relate?

What drives current variabilities in tempo? What impacts can individual differences, workers' roles, or specific organizational approaches have?

Where do conflicts sometimes occur due to misunderstandings around tempo? When and why do collaborators become "out of step"? Could these current problems present opportunities for your team's product?

What do individuals and organizations think of current tempos in targeted work practices? What parts of their work would they like to slow down or speed up? Why?

What positive or negative impacts might your sketched application concepts have on various tempos? What problematic changes seem sufficiently possible to imply that your team may want to redesign related functionalities?

How might the inherent tempos of your sketched functionality concepts be received by an aging knowledge workforce?

What interactivity and design communication could positively influence workers' perceptions of elapsed time during their experiences with your team's computing tool?

How might positive changes in targeted tempos factor into your product's brand?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D2. Expected Effort

Knowledge workers develop useful expectations about how much time and attention is required to successfully accomplish different operations, tasks and larger activities. Product teams can envision functionality concepts that could either meet or exceed these expectations, providing justifications of sufficient value whenever onscreen tools happen to require more work instead of less.

Examples from three knowledge work domains:

> An architect expects the act of grouping elements together within her building modeling application to be rapid and direct. She is surprised when she is prompted to specify seemingly unimportant information about a grouping before she can proceed with unifying it **(see illustration on next page)**.
>
> A financial trader "test driving" a new trading application expects the optimal pathways for certain common tasks to be as fast as they are in the tool that he currently uses. His opinions about novel functionality are more open.
>
> A scientist, while specifying the attributes of several clinical samples in her lab's information management application, is surprised by how quickly she is able to enter required and desired information for each sample.

Knowledge workers often develop strong opinions about the time and attention requirements of specific work practices (A). Workers' abilities to make accurate estimations of effort can be considered a valued part of their expertise. Additionally, experienced individuals have often become highly skilled at completing some operations and tasks, allowing them to invest much less effort in these actions than new practitioners (D7, K6).

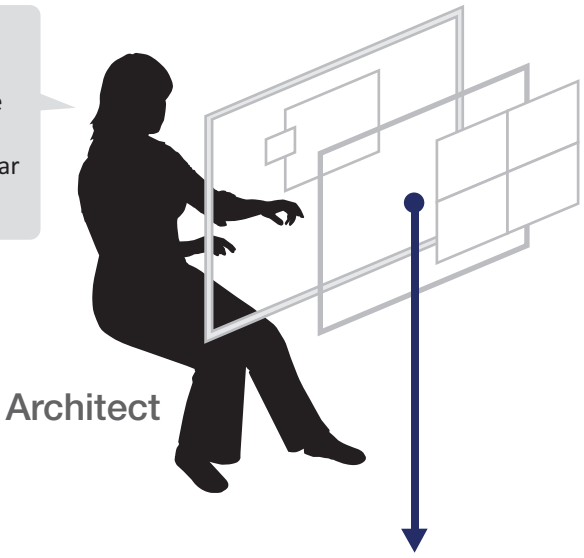
Product teams can strive to make the amount of effort that workers expend in an interactive application feel congruent to the benefits that a tool provides in their work practices. People may expect some elements of their work to be less effortful after adopting computing tools (E, K), especially in tedious tasks that they find less engaging and valuable in the context of their larger goals (D3, D4). When the characteristics of a functionality concept result in workers needing to expend more effort than expected, teams can attempt to reframe users' expectations by communicating the value of these additional efforts (C1, K2, K7).

When product teams do not actively consider workers' expectations of effort in targeted operations, tasks, and larger activities, resulting applications may contain interactions that users view as too difficult or demanding. Especially when extra effort does not provide understood and compelling value, workers may believe that these tools are based on a faulty understanding of their needs (A4, K3). They may also feel that time spent on inappropriately effortful tasks stressfully detracts from more important work outcomes (L1).

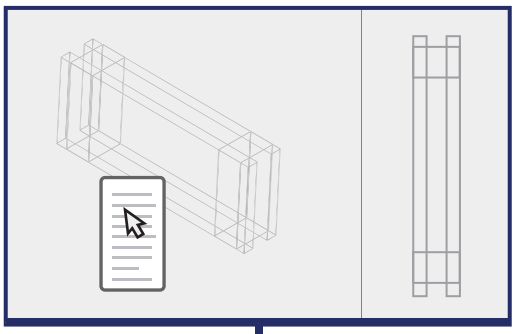
Conversely, applications can force too much streamlining of work, removing certain interactions and awarenesses that individuals enjoy or value in a practical sense (C6).

See also: B10, C4, C8, D, G, J1, K8, K9, M1, M4

I'm going to group some separate parts together into a single part, because I want them to always appear as one thing...

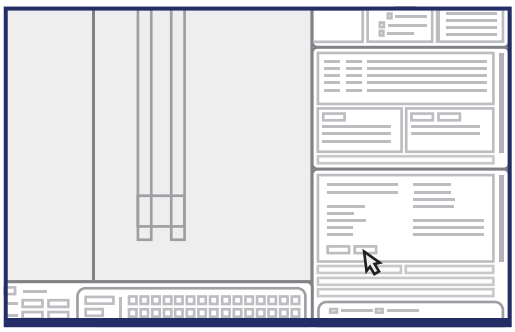


So I've got the pieces selected, and I'm opting to group them into a collective object in the building model...



And the application wants me to fill out all this info about different properties, which really doesn't seem necessary to me...

I guess that the info could be useful later, when I have these things all over the building model, but I'm not sure that it's worth doing those steps every time I want to create a new a grouped object...



Key *application envisioning* questions:

What expectations of effort do targeted knowledge workers have in the specific areas of work practice that your team is targeting? Which of your team's functionality concepts will likely "beat" those expectations? Which might be perceived as problematically effortful to use?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How much effort do targeted workers currently spend on the specific operations, tasks, and larger activities that your team is striving to mediate? What benchmarks do they use?

What do targeted individuals and organizations think of these current levels of effort?

Do people find any of their current work practices to be repetitive or tedious?

What practices do knowledge workers not want to change, despite high effort requirements? Why?

What general expectations do workers have about the impact of computing tools on the effort needed to accomplish their workplace goals?

How do expectations of effort vary across targeted individuals, roles, organizations, and other factors?

Which of your functionality concepts will likely be recognized as significantly reducing effort in certain activity contexts? Is this a compelling value proposition?

Where might workers accept additional effort in a new computing tool if it was seen as providing additional value? How could that value tradeoff be embodied and communicated in your sketched functional offerings?

What design approaches might make work feel like it is taking less effort than it actually is? What advanced analogies to other products and domains could inform your team's ideas about reducing perceived effort?

How might your scenarios for desirable reductions in effort factor into the story of your product's brand?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D3. Current Workload, Priority of Work, and Opportunity Costs

Knowledge work often involves pools of collected work items that can be generated by workers for themselves or can arrive via structured handoffs and other communications. Product teams can envision features that could support workers as they strive to understand their current workload, assign priorities, and then focus their efforts on certain items.

Examples from three knowledge work domains:

> A scientist views a list of all of the experiments that have been recently run for a clinical study, narrowing in on the items that require her approval in order for their results to be copied to her lab's analysis database. She scans the list and chooses to review samples from the most interesting experimental group first **(see illustration on next page)**.

> A financial trader visually scans a list of offers in his trading application. The trading day is almost over, and since he has been repeatedly distracted by some interesting incoming offers, he decides to work only on trades that match the priority list that his group made this morning, before the markets opened.

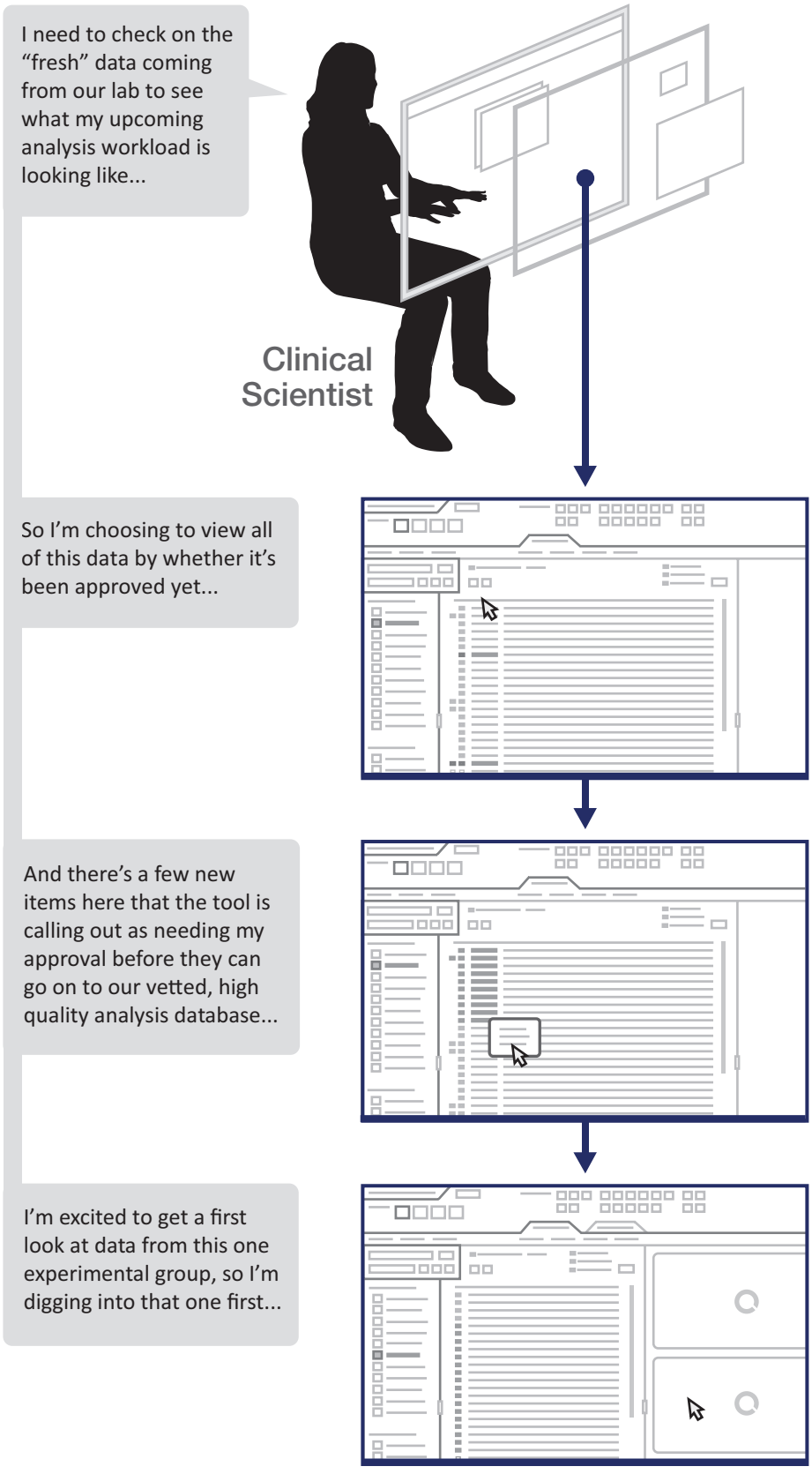
> An architect has been assigned a long list of areas in a building model that she needs to detail out within her building modeling application. She decides to get started on those areas of the draft model that other members of her team will be working spatially adjacent to soon, leaving a number of messages and notifications unviewed until she has made some initial progress.

Knowledge workers are often passionate about accomplishing certain goals in their chosen vocations. These goals can range from macro, extended visions to micro, day to day intents. When faced with time limitations and decisions about what work to accomplish next, individuals may prioritize their options and weigh the opportunity costs of taking certain courses of action (D1, D4). Alternately, they may choose their next task based on proven heuristics.

Product teams can envision functionality concepts that could valuably support awareness and critical decision making around users' workloads. For example, interactive applications can generate tailored information representations (E3, E4, F) that organize current work items (C5). These manipulable views (I2, I3) can increase the perceptual salience of time sensitive items and demote lower priority options based on their defined states (B5, B6). Once users choose a work item to pursue, applications can provide them with direct pathways to relevant actions (C4).

When product teams do not actively consider how their application concepts could influence knowledge workers' management of their own workloads, resulting tools can force users to spend additional time planning and tracking their efforts (D2). Without appropriate information displays, workers may overlook high priority needs, potentially resulting in timing errors and lost opportunities (C9, G3). Similarly, cooperative and collaborative work can also be affected when multiple workers struggle to understand the scope of work items that require, or could benefit from, their attentions (B7, C7, G4).

See also: A, D, E, G5, I4, K13, M1, M4


I need to check on the "fresh" data coming from our lab to see what my upcoming analysis workload is looking like...
Clinical Scientist
So I'm choosing to view all of this data by whether it's been approved yet...
And there's a few new items here that the tool is calling out as needing my approval before they can go on to our vetted, high quality analysis database...
I'm excited to get a first look at data from this one experimental group, so I'm digging into that one first...

Key *application envisioning* questions:

How might your team's functionality concepts allow targeted knowledge workers to assess the workload that is currently "on their plate," prioritize what they want to accomplish, hide or remove what they do not want to address, and work on selected items until their "plate is clean"?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently assess their workload while accomplishing the practices that your team is striving to mediate?

How do knowledge workers and organizations keep track of the larger picture of their collective activities, instead of focusing only on granular tasks?

Where do various work items arrive from? How do colleagues and collaborators stay aligned around each others' progress?

How do workers establish priorities? How do they assess the potential opportunity costs of addressing certain work items at the expense of others? Do these decisions follow established procedures or are they typically based more on impromptu judgments?

What breakdowns currently occur in these decision making tasks? Could these problems present opportunities for your team's product?

What factors can change the priority of a work item? How do people "shift gears" to address high priority work?

What currently happens to completed items in order to remove them from workers' proverbial "plates" so that they can focus on needs that have yet to be addressed?

What larger design trends and advanced analogies to other domains could influence your team's ideas about thoughtfully facilitating these decisions and actions?

What functionality concepts might your team sketch with the goal of supporting workers' existing practices for assessing workload, assigning priorities, and understanding opportunity costs?

What additional challenges and possibilities for managing workload could your computing tool present?

How might volumes of data be meaningfully displayed in ways that could allow workers to better focus their time and attention? How could defined object states serve as a basis for clearly communicating present work needs?

How might your team's ideas about comprehensible onscreen workloads relate to your other design responses for supporting work in the context of volumes of information?

How might your application concepts provide additional support in these areas for an aging knowledge workforce?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D4. Minimizing Distraction and Fostering Concentration

Knowledge workers are often interrupted from the immersive flows of their own practices, and some of these interruptions may undesirably pull them away from valued actions and outcomes. Product teams can envision their functionality concepts with the intention of minimizing unnecessary distractions and other obstacles to workers' concentrated engagement in their present goals.

Examples from three knowledge work domains:

> A financial trader quickly books a peak number of trades, using multiple communication channels in parallel. While he can always access what he needs to make these important deals, his trading application does not interrupt him with certain types of new information until he has completed a long series of trade forms **(see illustration on next page)**.
>
> A scientist performs early explorations of a large clinical data set in her analysis application. Since she is just getting a sense for the data's overall "shape," she selects a calm and minimal browsing mode that turns off certain dynamic features that she sometimes finds distracting.
>
> An architect has completed a set of construction details in her building modeling application, after working on them for a couple of hours. While she waits for the tool to merge her relevant local files with the master building model, several lower priority notifications, which had been withheld while she was actively working, appear on her screen.

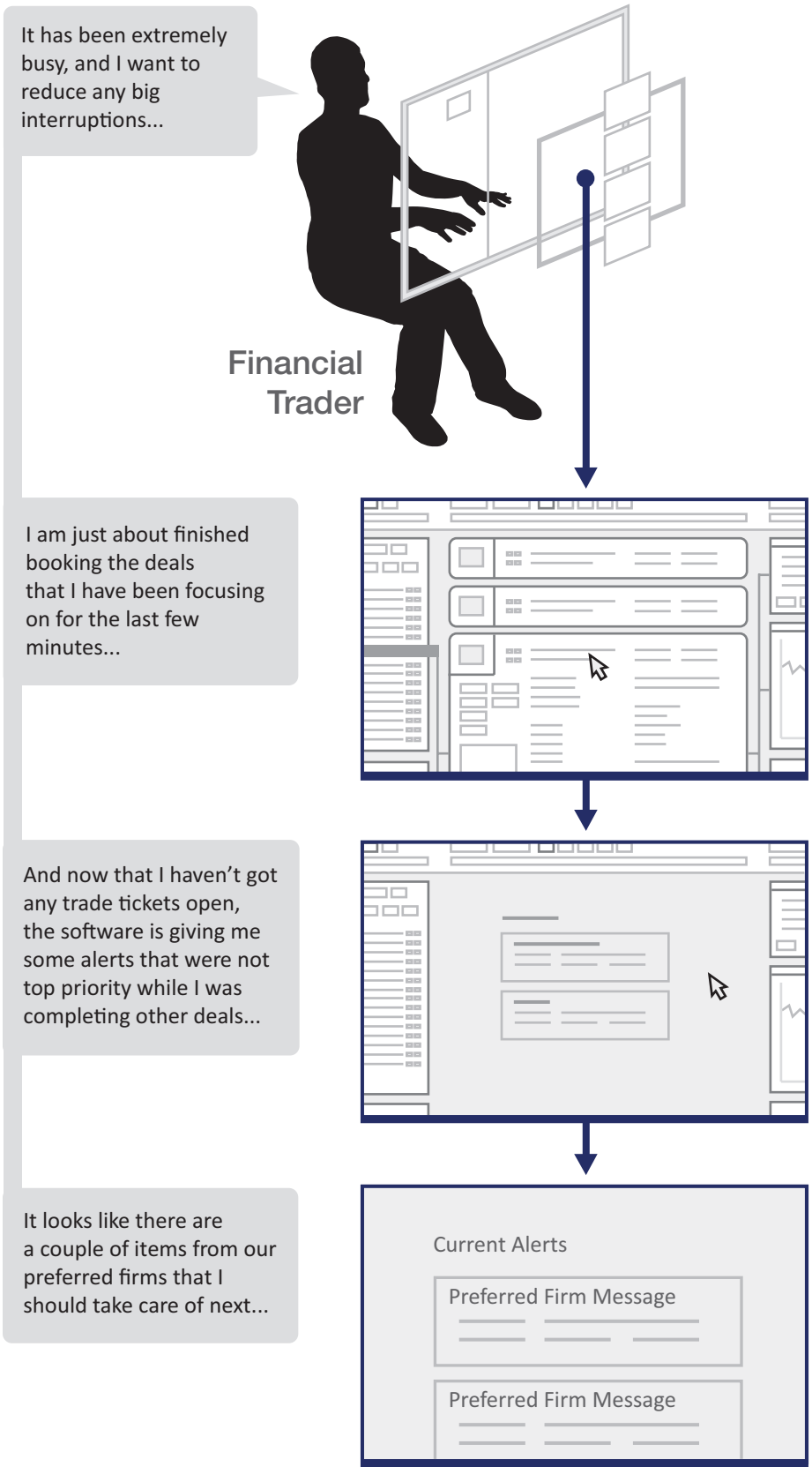
The multipurpose nature of many computing technologies creates opportunities for diverse distractions that can contribute to or interfere with people accomplishing their goals (A). While productive interruptions can include informal collaboration with colleagues (A7, C7, G4) and other timely communication (J), unwanted distractions can include uninformative messages (D6) and the sudden intervention of unpredictable processes (C1, G3, K5).

Product teams can identify parts of their sketched functionality concepts where certain distractions might be damaging. They can then envision defensive approaches that are tailored to these behavioral situations (D5, D6). As part of fostering concentration in attention intensive work, applications can promote the direct sense that workers' actions are tightly coupled to interactive results (B3, G1). This coupling can contribute to what the psychologist Mihaly Csikszentmihalyi has described as absorbing "flow experiences" (K13).

When product teams do not actively consider how their application concepts might encourage productive concentration, opportunities to promote focused and engaging user experiences can be lost. Resulting tools may contain a multitude of low value distractions that create ongoing stress (E6, D1), are difficult for workers to accommodate to (D2, D7), and can detract from the quality and quantity of work outcomes (D3, L1).

Conversely, if product teams take minimizing "unwanted" distractions too far, they may rule out high value functionality in the name of taming complexity.

See also: C8, C9, E, F9, K3, K6, K8, K10, M1


It has been extremely busy, and I want to reduce any big interruptions...
Financial Trader
I am just about finished booking the deals that I have been focusing on for the last few minutes...
And now that I haven't got any trade tickets open, the software is giving me some alerts that were not top priority while I was completing other deals...
It looks like there are a couple of items from our preferred firms that I should take care of next...
Current Alerts
Preferred Firm Message
Preferred Firm Message

Key *application envisioning* questions:

Where might your team's application concepts introduce unwanted distractions into targeted workers' practices? How could your sketched functionalities reduce unwanted interference while allowing for useful interruptions that may enhance productivity and quality in knowledge work?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What interruptions do targeted individuals currently experience in the work practices that your team is striving to mediate?

Which interruptions do knowledge workers value as contributing to their larger goals?

What distractions can have negative impacts on work outcomes? How strongly do people feel about these outside forces?

Which work practices can require intensive concentration?

Which tasks or larger activities currently allow workers to experience a satisfying sense of engagement under certain conditions?

Which interruptions frequently lead to observable errors or reduce the quality and quantity of workers' outputs? Could these problems present opportunities for your team's product?

What strategies do targeted workers currently use to try to minimize unwanted distractions?

How might your team's sketched application concepts influence workers' current experiences of distraction and engagement?

What undesirable distractions could your computing tool introduce? What approaches might your team envision to limit or eliminate these factors within your sketched scenarios for work mediation?

How might your application concepts promote and enhance existing forms of engagement in workers' experiences? How might your sketched functionalities desirably introduce this type of engagement into other practices?

How could your team's offering present calming "environments" for workers to act within, while at the same time usefully directing their attentions with relevant and appropriately weighted perceptual cues?

Where could interactions in your product meaningfully promote a strong sense of direct, tightly coupled connection with onscreen objects?

How might your application concepts provide additional support in these areas for an aging knowledge workforce?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D5. Resuming Work

Knowledge workers' activities often span more than one work day. Within a given day, individuals may shift their attentions back and forth among several different threads of work. To reduce the effort needed to effectively resume previous threads, product teams can envision useful cues that could prompt workers' recollections and outline current conditions within a shared workspace.

Examples from three knowledge work domains:

> A scientist arrives at her clinical lab and launches her analysis application. She selects the option to resume working on the last project that she had open, and the application displays every element of her view just as she had left it, including a reminder that she had entered to tell herself what to do next **(see illustration on next page)**.
>
> An architect logs out of her building modeling application so that she can attend a meeting, knowing that the exact same view, along with a message about her colleagues' current project tasks, will be called up when she logs back in after the meeting is over.
>
> A financial trader leaves an incomplete trade form open in his trading application while he books a more time sensitive deal. The half empty form serves as a reminder about the unfinished trade and allows him to quickly resume the task later.

It can be difficult for knowledge workers to "get back into" interrupted efforts, even after relatively short breaks. Relevant cognitive states do not reappear at the flip of a proverbial switch, though recognizable external cues can help workers to appropriately return their attentions to where they had left off.

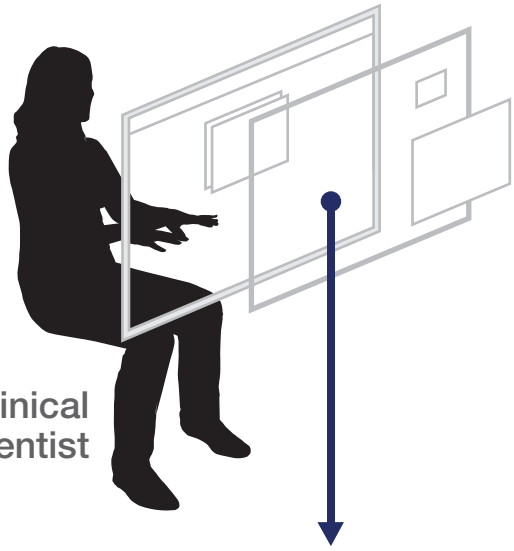
In order to envision valuable functional responses for resuming work, product teams can examine certain tasks and larger activities through the lens of potential interruptions (A, C5). At an application level, products can "remember" the contents and arrangement of a display exactly as workers had left it (E3, E4, C4). Saved display states can also reappear contextually, when users reopen particular interaction objects that they had previously modified, for example (B1, B2, G1). Alternately, workers can choose to save explicit "bookmarks" that they can later return to (E1, E2, H1).

Stored historical traces of cooperative action can also be useful when resuming work (H2, B5). For example, in cases where colleagues have modified shared interaction objects that were previously in use (B6, H3), workers may benefit from a concise update on relevant changes that have been made in their absence (C7, G4, D6).

When product teams do not actively consider how individuals might part from and return to different threads of knowledge work, resulting applications may force users to expend extra effort recalling and recreating where they had left off (D2, D3). In complex situations, people may make notable mistakes when attempting to get back into their previous states of focused attention (C9, G3). In response to these difficulties, individuals may resort to workarounds, such as creating external memory aids at interruption points (H).

See also: B7, D, E, H4, J4, J5, K13

After I've first sat down at the lab in the morning, I always look at our current data with fresh eyes to see if anything jumps out that I hadn't seen the day before...



So I'm logging into our lab's analysis application...

And the software opens as if I had never left it, waiting for me to hopefully have some big insight into these results...

Key *application envisioning* questions:

What could the experience be like when "stepping away" from, and then returning to, your team's computing tool? How might your application concepts support targeted knowledge workers as they seek to invoke and reconstruct their previous mindsets in order to "pick up" where they had left off in their evolving activity contexts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How often do targeted individuals currently step away from and return to the work practices that your team is striving to mediate?

What are some common scenarios for setting aside work? What do knowledge workers have to remember when resuming targeted tasks or larger activities?

How does the structure of their environments currently help them to recall "their place"?

What memory cues do they purposefully add for themselves? What other strategies do they employ to more quickly and accurately refocus their attentions on earlier threads?

What errors can occur when workers resume previous efforts? Could these problems present opportunities for your team's product?

What larger design and technology trends could influence your ideas about how your application concepts could support reconnection with work in progress?

How might your team's sketched functionalities reduce the difficulty of returning to earlier threads of work?

What application events, such as logging in or reopening a particular work item, could provide useful opportunities to implicitly recreate workers' views within your computing tool?

What specific interaction objects or application level elements could be restored in ways that may remind workers' of "their place?"

What explicit methods of bookmarking or otherwise cataloging work progress might workers find valuable?

How might shared uses of application content present opportunities to valuably highlight important changes as workers' resume their "paused" efforts?

How might your team's approaches for work resumption relate to your other concepts for supporting cognitive tracing, cooperation, collaboration, and workspace awareness?

How might your application concepts provide additional support in these areas for an aging knowledge workforce?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D6. Alerting and Reminding Cues

Knowledge work often involves event driven signals and actions, which the boundaries of computing displays may hide from an application's users. Product teams can envision timely and salient messaging that could reduce or eliminate the need for workers to continuously monitor for certain events that might impact the sequence or outcomes of their efforts.

Examples from three knowledge work domains:

> A financial trader, while analyzing a potential trade in his market information application, is presented with a message that reminds him that an earlier, unrelated offer is about to expire in his trading tool. The message also provides him with direct options to accept or decline the pending offer **(see illustration on next page)**.
>
> An architect receives a text message from her rendering application, which informs her that a lengthy image creation process has just been abandoned due to a critical error. She stops what she is doing to find a computer, log into the networked rendering server, evaluate the incident, modify some settings, and restart the process.
>
> A scientist receives an alert from her lab's information management application that all of the samples for a clinical study have been processed and that the study's experimental data can now be analyzed.

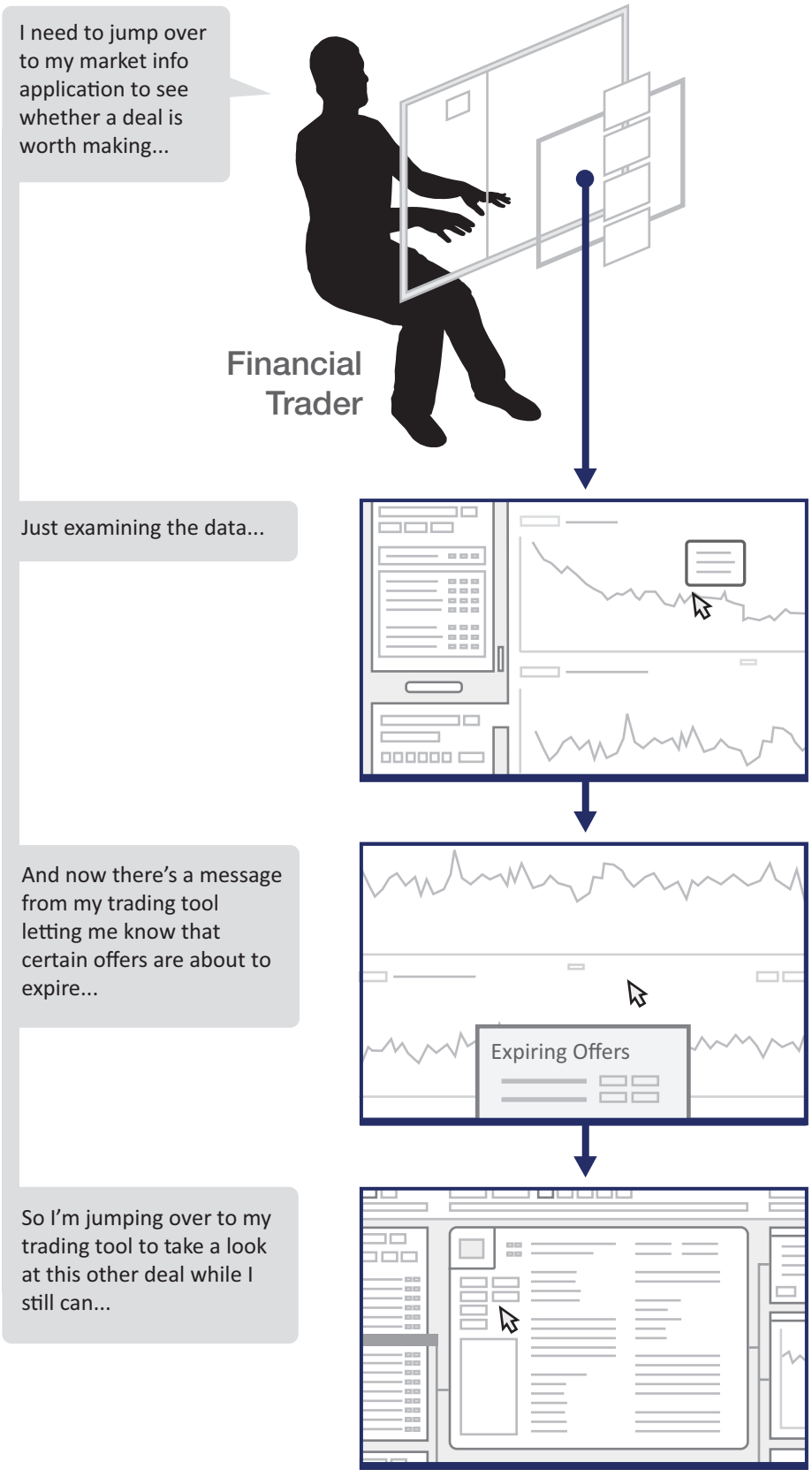
Actively attending to multiple threads of complex knowledge work at the same time — roughly speaking — can be mentally taxing, if not impossible. The ability to effectively monitor for key situations across more than one thread of work is often considered a useful and valued skill.

Interactive applications with features that intensively support collaboration (B7, C7, G4) or automation (E) can change the nature of what it means to attend to conditional and time sensitive events. Maintaining diligent attention under these circumstances can be difficult for a variety of reasons, generally rooted in the sense that progressive disclosure can often effectively "hide" important realities.

When it is not essential for knowledge workers to actively monitor a process, product teams can envision concepts for automated (E3, E4) alerting and reminding cues. Relevant, visible, and timely messages can usefully reduce or eliminate the need to remain vigilant for certain application (C10) or object states (B5, B6). Similar to appropriate error messaging (C9, G3), teams can generate these notifications from a strong understanding of workers' goals, avoiding unnecessary distraction (D4) and providing direct access to related actions (G1, C4).

When product teams do not actively consider how their application concepts could offload attentional effort through alerts and reminders, resulting tools may require workers to persistently attend to the presence or absence of certain cues in order to efficiently transition through their practices (A). Since these automated messages are commonly included in many genres of computing tools, workers may find vigilance tasks without these triggered notifications to be tiring and unnecessary user experiences (D2, D3).

See also: C5, C8, D, E, H3, K7, K13


I need to jump over to my market info application to see whether a deal is worth making...
Financial Trader
Just examining the data...
And now there's a message from my trading tool letting me know that certain offers are about to expire...
Expiring Offers
So I'm jumping over to my trading tool to take a look at this other deal while I still can...

Key *application envisioning* questions:

What events in your team's application concepts will targeted knowledge workers likely want to know about and monitor for, either as insight into mediated work process or as event driven support for their own memories over time? How might the automated presentation of relevant messaging allow users to stay attuned to these events without maintaining vigilant attention for them?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently remind themselves of important, time sensitive information in the work practices that your team is striving to mediate?

What conditions do they monitor for in their activity contexts?

Which of the automated functionalities that your team has envisioned could potentially benefit from alert and reminder options? How might these options provide value by reducing or eliminating effort that would otherwise be needed to attend to your product's workings?

What information and conditional events in your application concepts might workers like to have reminders about over time? How might they set up these memory supporting messages?

What relative priorities could be appropriate for the different types of alerts and reminder messages that your team has sketched?

How directive should various types of messages be? Which could be strictly informational? Which might workers need to actively address within a certain timeframe?

How might lower priority alerting and reminding cues be presented at transitional "seams" between attention demanding tasks and larger activities?

What communication channels could be most effective for delivering different types of alert and reminder content? Is notification within your computing tool enough?

How could interrupting messages present related pathways of action so that targeted workers do not need to locate relevant navigation options?

How might individual users customize their own alerts and reminders to call out those events that they value and to ignore those that they do not?

How might these messages be experienced within groups of cooperating or actively collaborating workers?

How might your team's approaches for alerts and reminders relate to your other concepts for supporting cognitive tracing, cooperation, collaboration, and workspace awareness? How could these messages relate with your product's error prevention and handling conventions?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

D7. Eventual Habit and Automaticity

Over time, knowledge workers learn to attend to certain areas of their interactive applications, while deemphasizing other pathways and content. Product teams can sketch their functionality concepts with this sort of habitual learning in mind, creating conditions where workers may develop adaptive, nearly automatic approaches to accomplishing routine interactions.

Examples from three knowledge work domains:

> An architect works through a cascade of dialogs to change a very specific setting in her building modeling application. When she first used the tool, this navigation seemed excessively effortful. Now, she does "not even think of it" as she performs the task seemingly "automatically" **(see illustration on next page)**.

> A financial trader expertly tabs through the fields in a trade form, entering specific data and making relevant selections. To help him move on to his next trade more rapidly, he is in the habit of selecting an option that books a completed trade and automatically opens an empty trade form.

> A scientist, having learned a preferred pathway for narrowing in on subsets of valuable data in her analysis application, quickly moves through a series of complex visualizations in a specific sequence.

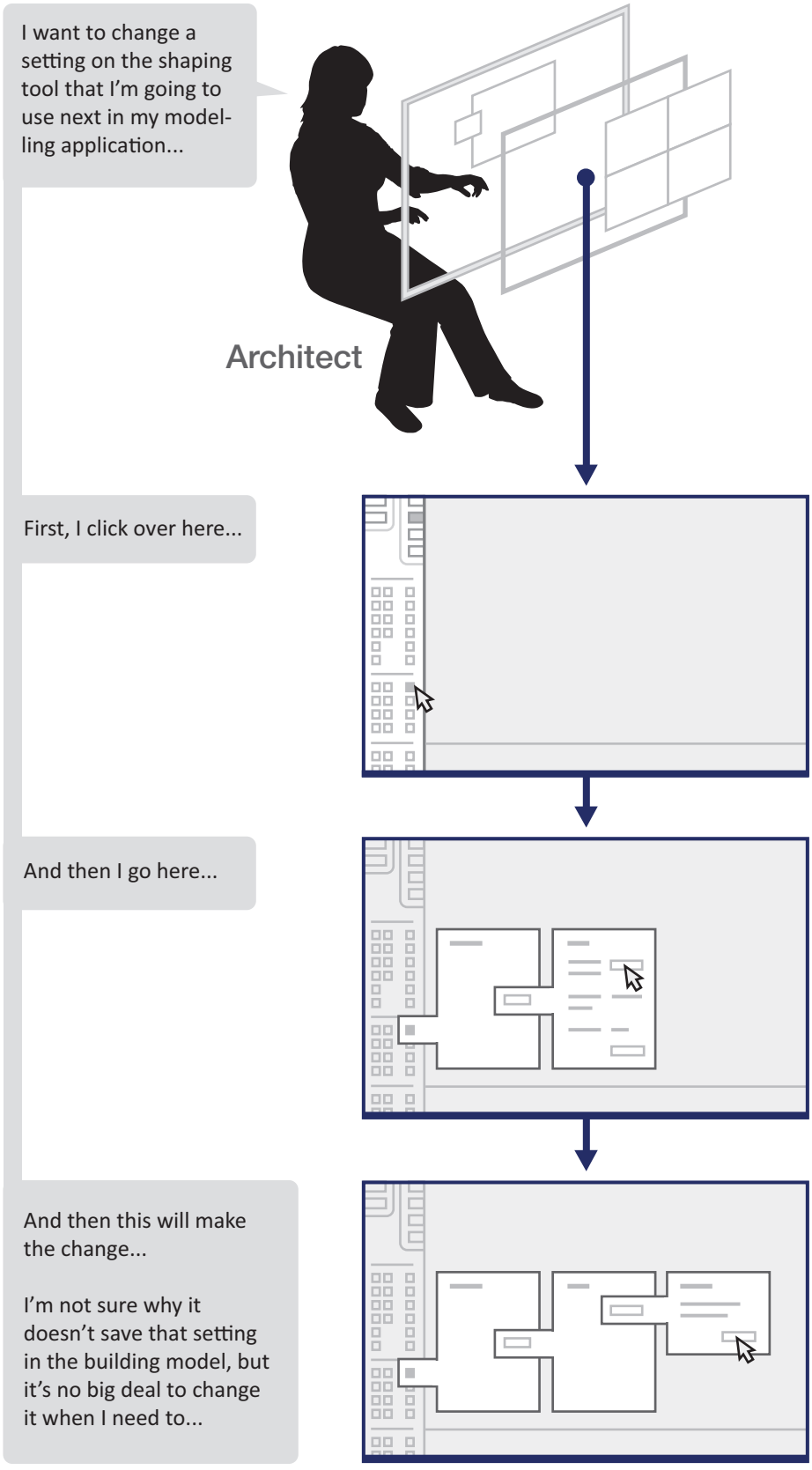
Knowledge workers' can show surprising skills for incorporating new artifacts into their work practices. Even in cases where individuals do not recognize that they have these abilities, people may use less and less of their conscious attentions as they repeatedly act on or with new artifacts in specific activity contexts (A, C4).

In the same vein, while initial interactions (K2) in a new computing tool may demand workers' intensive attentions (D2, D3, D4), over time, people can develop varying levels of adaptive habits within routine and relatively unvarying pathways. In some situations, highly entrenched habits can develop into automaticity, meaning that specific operations or larger tasks (A5) may eventually require limited conscious consideration on the part of application users.

With these innate human tendencies in mind, product teams can identify areas in their sketched design concepts where interactions are likely to be frequent and mental efforts are likely to decrease due to consistent goals and the crystallization of standard approaches (A3, A4). Teams can then refine these functionality concepts with the goal of promoting workers' acquisition of adaptive, tacit abilities. These refinements can include, for example, clear and direct narratives of interaction (G1), uncomplicated conceptual models (C1), and appropriate instructional frames (K2, K5, K6, K7).

When product teams do not actively consider how workers might develop habits and automaticity in their application concepts, opportunities to facilitate certain forms of mastery in users' experiences can be lost. Resulting products may put too much emphasis on initial learning rather than accommodated usage, potentially leading to the development of negative habits for the long term (K5). Workers may also experience severe frustration when updated applications are not built from an understanding of their "legacy" of learned adaptations (M1).

See also: A, C8, D, E6, K8, K12, K13


I want to change a setting on the shaping tool that I'm going to use next in my model-ling application...
Architect
First, I click over here...
And then I go here...
And then this will make the change...
I'm not sure why it doesn't save that setting in the building model, but it's no big deal to change it when I need to...

Key *application envisioning* questions:

Assuming that targeted knowledge workers will eventually adopt and frequently use your team's computing tool, how might you examine your application concepts through the lens of users' eventual habituation and mastery? What unpredictabilities could lead to errors by "getting in the way" of valuable automaticity? Where might negative habits develop?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Where have targeted individuals already developed useful habits and automaticity in the tasks and larger activities that your team is striving to mediate?

What errors currently occur due to knowledge workers "automatically" acting in inappropriate ways? Could these problems present opportunities for your team's product?

How might your sketched functionality concepts meaningfully reference workers' existing, productive habits?

Where in your application concepts might targeted workers develop new habitual behaviors after frequent use of certain options?

Where could work practices mediated by your computing tool be repeated and consistent enough for workers to attain a degree of useful automaticity?

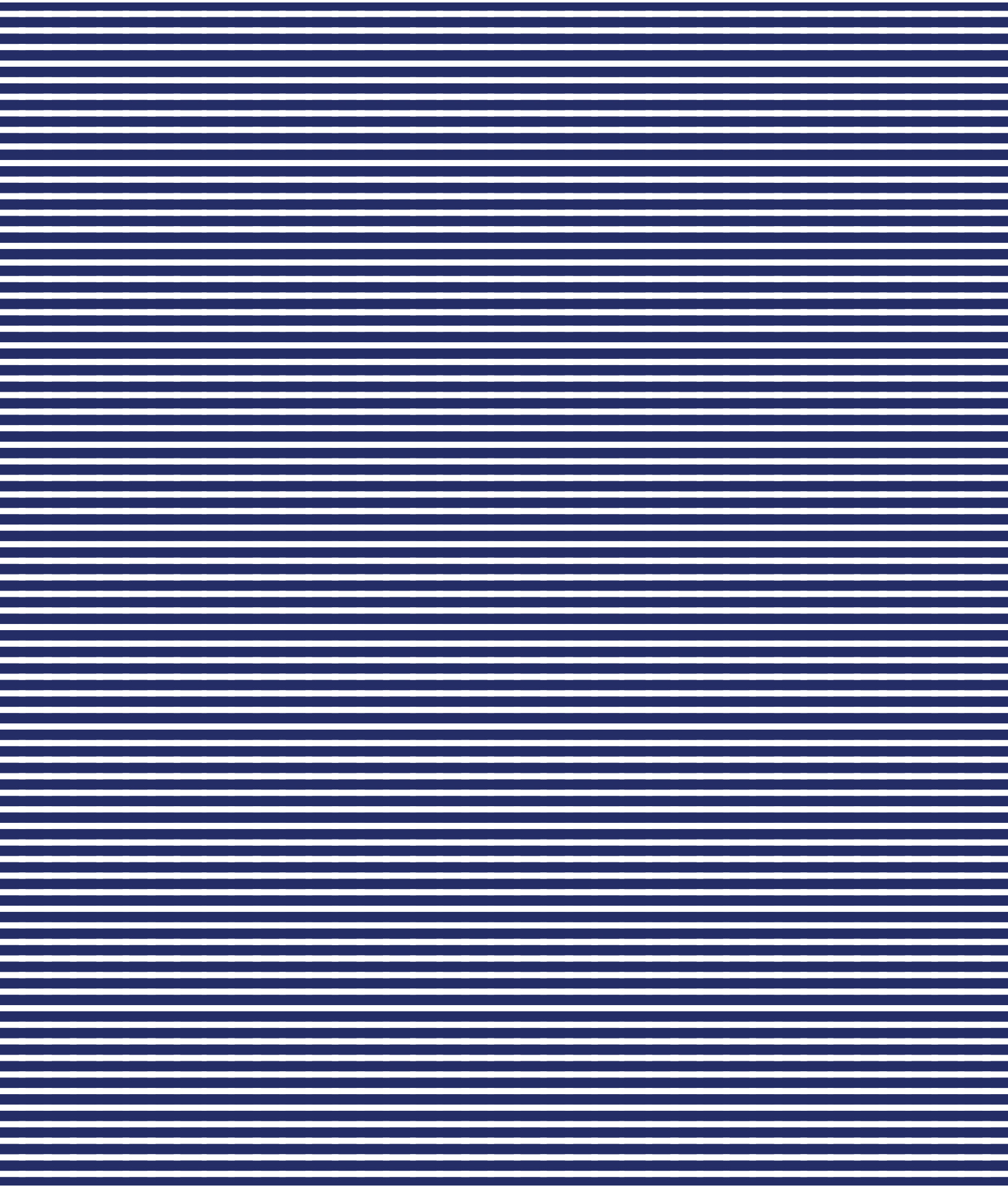
How might certain predictable behaviors in your functionality concepts allow individuals to quickly navigate their frequent interactions in increasingly "effortless" ways over time?

What negative habits could workers form within the channeling flows of your sketched application offerings?

What errors might stem from users automatically interacting onscreen out of entrenched habit instead of considering the unique characteristics of their current situations?

What design responses might your team envision to reduce or eliminate certain opportunities for negative adaptations and automaticity errors? How might these methods tie into your larger error prevention and handling approaches?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?



E.

Providing Opportunities to Offload Effort

Valued computing tools can desirably reduce burdens in knowledge work while at the same time promoting a sense of engagement and agency.

Designing for such useful reductions requires a deliberate and critical understanding of current and potential efforts in work practice.

During *application envisioning*, product teams can map workers' consistent and routine burdens in order to locate potential opportunities for supporting technologies.

By focusing on how effort might be offloaded to an onscreen tool, teams can highlight cases where higher order tasks and user experiences might transformatively replace unwanted actions and cognitive load.

All of us face limitations in what we can accomplish. There are only so many hours in a day, and our human minds can, roughly speaking, only process or actively remember so much at any one time.

Knoweldge workers make use of valued tools to get more done and to make their lives feel simpler. People can become adept at arranging and manipulating the world around them to make their actions easier, thereby improving their ability to accomplish certain outcomes. By appropriating useful artifacts into their practices, individuals and their organizations can positively transform work that would otherwise require tedious labor or complex mental operations.

Product teams can envision opportunities for knowledge workers to distribute effort among themselves, their colleagues, and their computing tools. High level ideation around "what people are good at and what computers are good at", while useful, may not drive teams to sufficiently consider the particulars of workers' specialized motives and local cultures. To arrive at powerful and valuable offloading options, teams can focus on possible intersections of specific burdens in work practices and potential technology responses that could either alleviate these burdens or augment workers' related abilities.

This category contains 6 of the 100 *application envisioning* ideas in this book:

E1. Offloading long term memory effort

E2. Offloading short term memory effort

E3. Automation of low level operations

E4. Automation of task or activity scenarios

E5. Visibility into automation

E6. Internal locus of control

Product teams can use these ideas to explore potential transformations of work practice through the reduction of specific memory burdens and appropriate automation of operations, tasks, or larger activities. Aging workforces within a product's demographics, who may be experiencing decreases in some of their faculties, may find such offloading options to be especially valuable.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Considering workers' attentions" (D), "Supporting outcome exploration and cognitive tracing" (H), and "Facilitating communication" (J) categories.

E1. Offloading Long Term Memory Effort

Certain information often needs to be "remembered" for some time by knowledge workers and their organizations. Product teams can envision functionality concepts that could record and store this valued content, allowing workers to refer to their computing tools instead of having to concentrate on keeping certain items mentally available.

Examples from three knowledge work domains:

> A scientist opens a file in her analysis application that contains data from a previous clinical study. Since the old study shares some similar parameters with her current work, she reviews the stored information to remind herself which analysis processes had previously led to valuable insights **(see illustration on next page)**.
>
> An architect, when faced with a problem in her current work, opens up older versions of the same building project in her building modeling application. She uses the stored information to help her remember how she had worked with civil engineers to resolve similar issues in their past.
>
> A financial trader uses his trading application to view his group's deals from yesterday so that he can see how much business he did with a particular entity. Without the tool's stored record, he would probably only be able to recall a few of the bigger ticket transactions.

Knowledge workers can face daunting memory burdens as their activities progress over extended periods of time (A). Luckily, people are not typically expected to recall everything; established work processes (A4, C6) and cultural norms (A1) often implicitly or explicitly acknowledge the strengths and weaknesses of our long term memories. These accommodations can be especially visible when work activities revolve around high volumes of information rich artifacts (B1, I) or reference large and constantly evolving information resources (I5, G6).

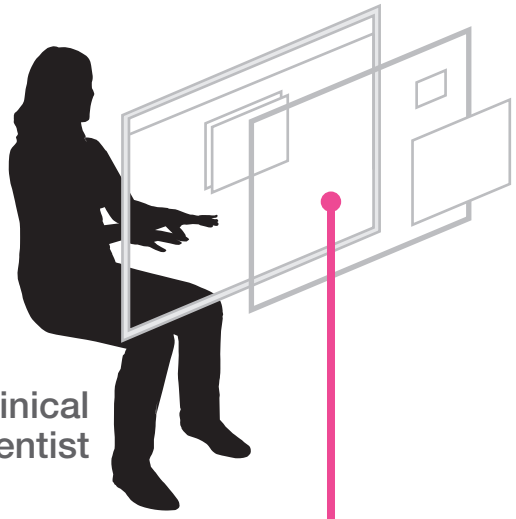
Since computers can excel at storing specifics, product teams can envision functionality concepts that could allow workers to record, locate, and recognize valuable information rather than attempting to engrain it in, and then recall it from, their long term memories (D4). Application functionality can usefully and meaningfully integrate existing forms of externalized long term memory (F2, G4, J2) that have historical trajectories within organizations and larger professions, such as online data repositories or the formats of certain paper records (J7, H1, H3).

Additionally, team's concepts for collaboration oriented features can indirectly help workers to distribute their remembrance efforts by enabling them to more easily reach out to colleagues' for their recollections (B7, F1, J5, H3).

When product teams do not actively consider how their application concepts could influence workers' long term memory burdens, opportunities to valuably reduce or eliminate certain types of unwanted memory effort can be lost. Resulting products may increase possibilities for recollection error (C9, G3) or force workers to create and enact effortful work arounds in order to prevent information from becoming "lost" (D2, D3).

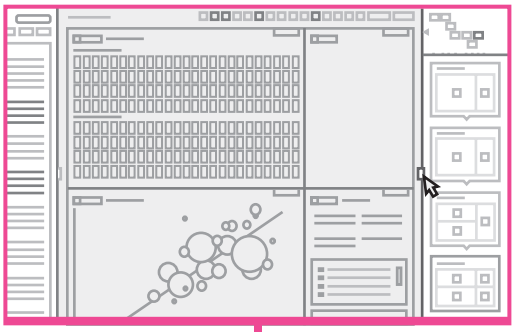
See also: B6, D, E, H, J4, M1, M4

As I'm looking at the data from our latest study, I'm vaguely reminded of how we analyzed the data from a big study last year...

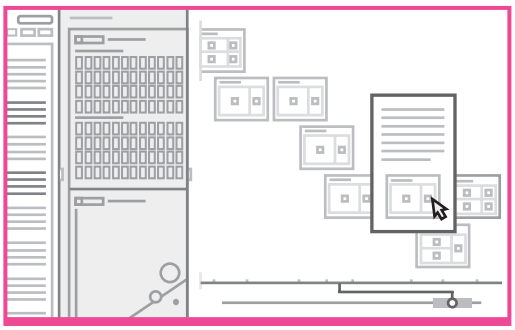


So I'm just going to open up that older study in our analysis application...

And look at the analysis history to see what settings and processes we used back then on this massive pile of results...



Oh, that's right, I had forgotten that we did it that way. Great. That same approach should be useful in our current round of work...



Key *application envisioning* questions:

What information do targeted knowledge workers struggle to remember over extended periods of time in the work practices that your team is striving to mediate? How might your application concepts structure, collect, preserve, and present valued long term information in accessible and meaningful ways?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently record and keep track of information that they would otherwise need to recall from their own long term memories?

What artifacts do knowledge workers create in order to offload their memory efforts and make information available to multiple people over time?

When do workers turn to these artifacts? What role do they play in targeted operations, tasks, and larger activities?

What long term recollection errors are common? Could these problems present opportunities for your team's product?

How much emphasis do individual workers and larger groups place on the creation and maintenance of collective, organizational memories?

What larger design and technology trends could influence your team's ideas about how your computing tool might offload certain long term memory efforts?

How might existing processes for personal and organizational memory be incorporated into your sketched functionality concepts?

Which memory cuing features of existing artifacts could be enhanced within your application's displays? How might your team tailor the representations of certain interaction objects in order to support workers' own memory strategies?

What new data in your application concepts could lead to new sources of memory load? How might your product usefully record and present this content in ways that could alleviate these potential burdens?

During what tasks and larger activities could people benefit from being able to easily and directly access relevant stored information? What might these access points and pathways look like in your sketched functionality concepts?

What life expectancy could different types of stored information have? Could stored content ever become a hindrance or source of clutter in workers' activities?

How might your application concepts provide additional long term memory support for an aging knowledge workforce?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

E2. Offloading Short Term Memory Effort

Knowledge workers' short term memories have inherent limits, even in the context of familiar work practices. To support key short term memory challenges in computer mediated work, product teams can envision concepts for persistently presenting workers with recent cues and information that is pertinent to their goals.

Examples from three knowledge work domains:

> A financial trader uses a shorthand function in his trading application to enter key information about a list of deals that he is negotiating on the phone. After the call is complete, he is able to transform his quick notes in the shorthand function into a set of separate, fully detailed and booked trades **(see illustration on next page)**.
>
> A scientist zooms in on a progressively narrower set of clinical data in her analysis application. After spending a moment inspecting a small grouping of data at a very granular level, she quickly zooms out to remember which region of the clinical results set she was looking at. A small box traces the previous zoom area within its larger context.
>
> An architect stops what she is doing in her building modeling application to quickly place drafts of three structural features. She then flags each placeholder feature as work in progress, which changes them to a recognizable color. The presence of these colored volumes in her view reminds her what she wants to work on next.

We all work from the understanding that people can only actively maintain so much new information at once. The limitations of short term memory are a well characterized aspect of human cognition. Although knowledge workers can become skilled at keeping domain information at the forefront of their thoughts, they may also develop opportunistic approaches for using external resources to mitigate their inherent memory limitations (A). For example, workers may keep relevant information "near to hand" by printing important screen contents (J7), leaving useful documents open (G5, F1, F2), and writing shorthand notes while they work (H4, J5).

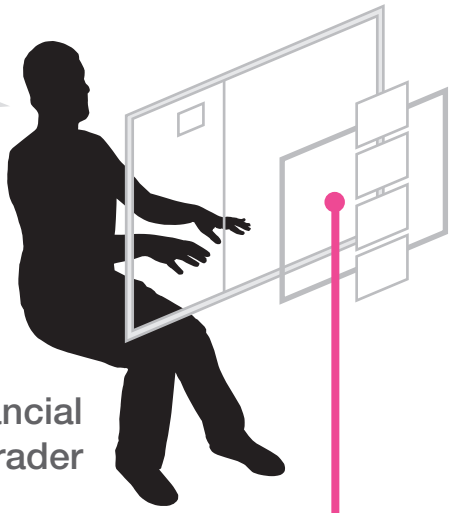
Product teams can envision functionality concepts that could support workers' desires to their offload short term memory efforts (C3). This support can also take the form of targeted refinements of existing functional options. For example, applications can provide fast access to recent information either through continuous display or by on demand access via clear interaction pathways (C4, F9, G6).

When product teams do not actively consider how their application concepts could influence workers' short term memory burdens, opportunities to valuably reduce or eliminate certain types of unwanted memory effort can be lost. Resulting products may promote possibilities for error in recall (C9, G3) or force workers to create and enact effortful work arounds in order to prevent information from becoming "lost" (D2, D3).

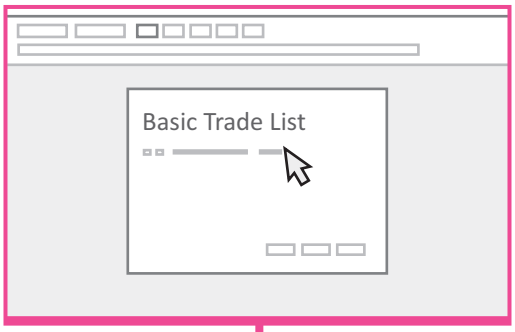
Conversely, explicit functionality and design responses in support of short term memory can be limiting or distracting (A9, D4), especially in cases where teams do not consider progressive disclosure of recent content as viable support.

See also: B2, D, E, F8, G4, H, J2, M1, M4

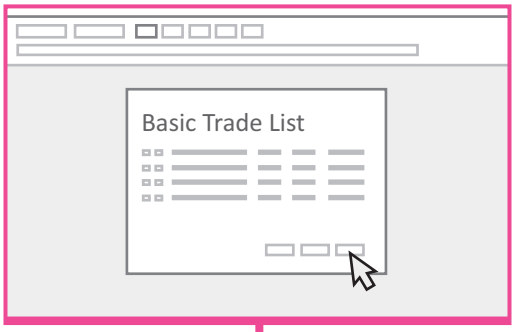
Often, there are too many deals made in a single phone call to remember them all without somehow getting them down on paper or my screen...



So I try to type them into this shorthand function, designed specifically to help with big lists of potential deals...



Then I can select an option to turn them all into full fledged trade tickets, which automatically makes some assumptions and fills in a lot of the information...



Once they are turned into individual tickets, I can review the information on each one, make any changes that I want to make, and then complete each deal separately, like normal trades...

Key *application envisioning* questions:

What information do targeted knowledge workers struggle to remember for short intervals while accomplishing the operations and larger tasks that your team is striving to mediate? How might your application concepts store and display relevant short term information in accessible and meaningful ways?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What strategies do targeted individuals currently use to keep track of operative information that they need to have mentally available or effectively "nearby" to successfully accomplish their work?

What artifacts do knowledge workers create in order to offload their short term memory efforts? How transitory are these objects?

What types of "active" information do targeted workers often forget when they are interrupted?

What larger design and technology trends could influence your team's ideas about how your computing tool might offload certain short term memory efforts?

Which memory cuing features of existing artifacts could be enhanced within your application's displays? How might your team tailor the representations of certain interaction objects in order to support workers' own memory strategies?

Where might navigation through your sketched functionality concepts introduce new short term memory load? How might persistently presenting recent and relevant information reduce or eliminate some of these burdens?

What functionality concepts or smaller design responses might your team envision to allow workers to explicitly record or highlight specific information that they want to remember in the short term?

What programmatic methods could valuably identify categories of "active" information and abstractly indicate these items with compact and learnable cues?

How might your team's concepts for supporting individuals' short term memory influence common ground and collaboration is shared workspaces?

What life expectancy could different types of short term information have? When could the persistent presence of this supporting content become a hindrance or source of clutter in workers' activities?

How might your application concepts provide additional short term memory support for an aging knowledge workforce?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

E3. Automation of Low Level Operations

Knowledge workers may experience certain frequent, highly granular work operations as redundant or excessively rigorous. To reduce or eliminate efforts around certain tedious or exacting operations, product teams can envision small, highly targeted automations within their sketched functionality concepts.

Examples from three knowledge work domains:

> An architect's cursor snaps to the edge of a form that she is trying to enclose in her building modeling application. Since she is familiar with the tool's behaviors, she anticipates the correction and, as a result, spends less time positioning her cursor accurately **(see illustration on next page)**.

> A financial trader is booking a deal in his trading application. As he fills in data, the application predicatively defaults subsequent fields, which he then simply tabs through if he agrees with the values that the system has entered.

> A scientist selects a different filter for a graph within her analysis application, and the transformed representation of clinical data instantly appears. Without the application's automation of the graphing operations needed to update this display, the resulting transformation would have taken significant time and effort to manually complete.

The term "computer" is famously derived from the specialized job that the technology initially replaced — the now extinct profession of manually computing mathematical problems for science, engineering, and business needs. Since that time, developments in computing have only extended this founding notion of offloading well characterized and predictable operations in knowledge work (A4, A5).

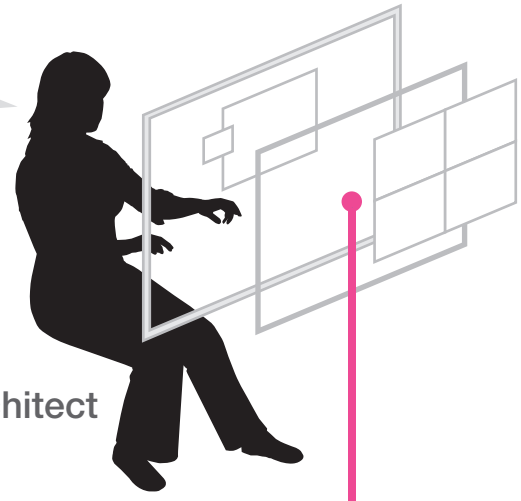
Product teams can envision how their interactive applications might augment specific work practices by performing small, useful, and learnable optimizations in the context of users' actions. To ensure that these small interventions are visible and understandable (E5), computing tools can provide cues to indicate where automations have occurred, as well as how their effects may be removed (C4, D6, H2). Depending on workers' expectations of control (E6), these granular automations can be the subject of customization choices (C8, K11).

When product teams do not actively consider how small operations in knowledge work could be usefully automated, opportunities to reduce workers' efforts (D2, D3) and to prevent certain types of errors (C9, G3) can be lost. Depending on their previous experiences with other computing interactions, workers may see the absence of some small automations as annoying oversights in a product's design (M1).

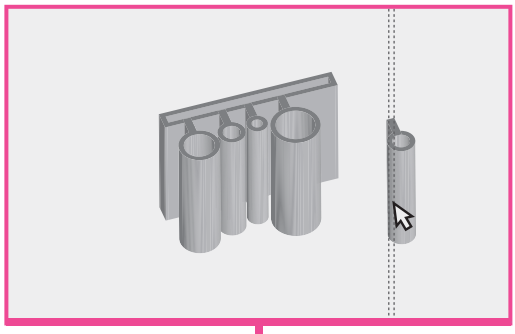
Conversely, in many cases, these small automations simply cannot be meaningfully envisioned due to broad variabilities in targeted work practices (A6, A7, A8). When misapplied, automation of operations can become a frustrating hindrance to the experience of directness in computing interactions (D4).

See also: B5, C10, D, E, I, M

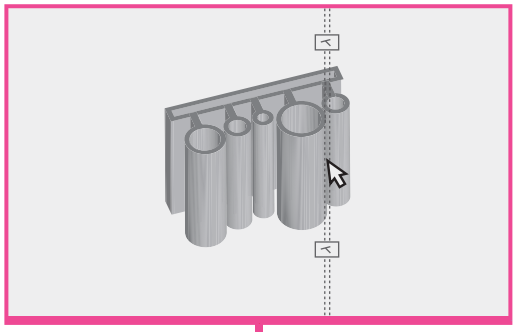
I'm finishing this shape, which I want to try out as a repeating motif on the facade of our latest building model...



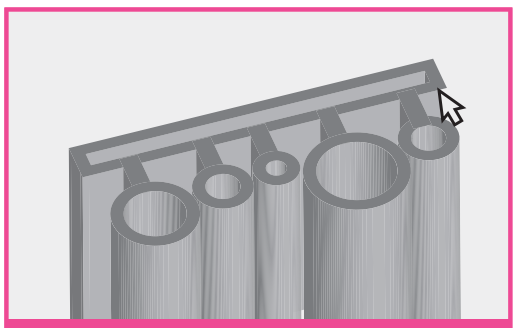
I'm selecting the unconnected form and dragging it toward the edge of the rest of the component...



And when I get close, it jumps to the surface in order to connect them...



I can override that small snap, but in this case it makes things a bit easier, and I know that they are truly connected...



Key *application envisioning* questions:

How might your team's functional offerings remove or scaffold certain consistent, granular knowledge work operations with highly specific automations? How could these small automations advance targeted workers' larger, goal directed tasks in useful ways that they may not even recognize?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What discrete operations in the work practices that your team is striving to mediate are standard, exacting, and tedious? What do targeted individuals think of these operations?

Where might your team's sketched functionalities introduce new operations that could also fit the standard, exacting, and tedious description?

Which operations in your concepts for work mediation might be usefully automated under the general goal of reducing users' efforts?

What larger design and technology trends could influence your team's ideas about small automations in your computing tool?

What predictive actions, useful suggestions, slight corrections, and refined interface tailoring could your application concepts automatically provide?

How might these automated operations reduce the incidence of predictable errors and corrective interactions? How could the design of these features relate to your products' larger error prevention and handling approaches?

Could certain small automations benefit from clearly communicated conceptual models, or could some of them provide just as much value if they are typically overlooked?

How might your envisioned automations impact workers' sense of control?

In what cases might targeted individuals see these automations as unpredictable or distracting nuisances?

What interaction methods could allow users to recognize and override the effects of certain automations?

What settings and customization functionality can your team envision to help ensure that automations will operate in accordance with workers' goals? How could these settings be clearly and contextually accessed?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

E4. Automation of Task or Activity Scenarios

In certain situations, entire tasks or larger activities in knowledge work can become extremely routine, describable, and tedious. In response to these cases, product teams can envision concepts for targeted automation functionality, which can change the nature of work by allowing individuals to focus more of their efforts on less routine and higher value efforts.

Examples from three knowledge work domains:

> A scientist designs a workflow in her lab's information management application. In this workflow, lab technicians will feed prepared samples into laboratory robotics, which will automatically gather experimental data. Her lab's computers will then automatically perform a number of algorithmic transformations on the data before storing the results in a repository where she can then analyze it graphically **(see illustration on next page)**.

> An architect enters parameters for the beginning and ending of a curved shape in her building modeling application. The computing tool extrapolates the entire surface of the form, including some of its engineering and construction details, based upon a set of customized functional rules and defined material properties.

> A financial trader books a transaction in his trading application and then immediately focuses his attention on his next potential deal. Behind the scenes, a whole series of crucial small tasks are automatically processed across a number of systems to make the completed transaction a reality.

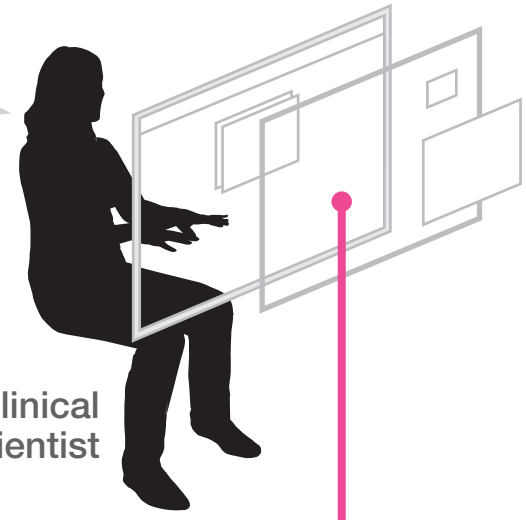
Historically, automation was an early focus in the application of computing to many workplaces. Today's product teams developing knowledge work tools may find that valuable opportunities for extensive automation of existing work practices (A) are not especially prevalent in the markets that they target. In certain cases, however, customizable (C8) automation of tasks or larger activities can provide transformative value in the context of workers' status quo practices (A9) and overarching organizational goals.

When product teams do not actively consider how larger units of work practice might be usefully automated, opportunities to reduce or eliminate unwanted effort (D2, D3), prevent certain types of errors (C9, G3), and drive precise, high quality outputs (A4) can be lost. Adopting highly "manual" applications may lead to people spending the same amount of time, or even more time, on less desirable, "lower level" work and user experiences. These "lower level" actions are often accomplished at the expense of other tasks that may better contribute to workers' desired outputs (L1) and larger goals (A5).

Conversely, when misapplied, larger scale automations can erode individuals' sense of control (E6) and drive corrections and workarounds that may require more effort than doing work without automated support (D4). Workers may place a high value on how they currently accomplish the tasks and larger activities that product teams perceive as prime candidates for automated offerings (A4, C6, E5). Even in cases where people desire larger scale automations, targeted work practices may contain prohibitive requirements for flexibility (A6, A7, A8).

See also: C10, D, E, F6, I, K4, K10, M

After my lab technicians prepare samples and put them in certain instruments, our lab's automation can do a remarkable amount on its own, with human eyes only on errors and exceptions...

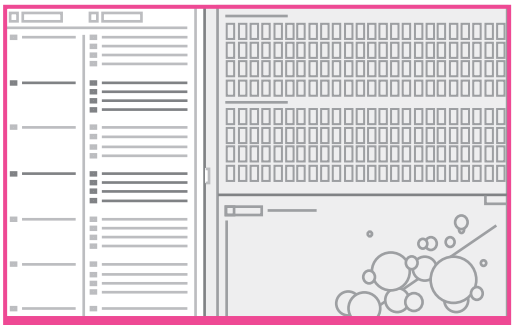


Clinical Scientist

LAB AUTOMATION CONTROLLED BY COMPUTING APPLICATIONS

- Automated data collection
- Automated data filtering
- Automated movement of data in study repository
- Automated calculation of resultant values in study
- Automated testing against previously coded hypotheses
- Automated messaging about data availability

And at the end of the automation pipeline, if all goes well, I receive accurate new data from the experiments I defined long before any of the lab work was even started...



Key *application envisioning* questions:

Is your team targeting any tasks or larger activities that have highly predictable and standard series of operations? What functionality concepts might you envision to automate these sequences? What could be gained or lost, from the perspectives of targeted knowledge workers and their organizations, in the adoption of such expansive automations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which of the work practices that your team is striving to mediate could be rationalized to the extent where automation may be a feasible option?

What tasks or larger activities, in practice, present "too much" variability for such functionality to be effectively defined and used?

Which work processes do targeted knowledge workers find tedious and time consuming? How do these routine processes currently distract from more meaningful and higher order pursuits?

What established processes do workers value in their current form, without automation? Why?

Which processes in your sketched application concepts might be usefully automated under the general goal of reducing users' efforts? What value could targeted organizations gain from extensive automations in the context of their larger goals and overlapping activities?

How might automated processes impact targeted workers' desired sense of meaningful visibility, direct control, and self determining agency?

What larger design and technology trends could influence your team's ideas about substantial automations in your computing tool?

How might the strengths of computing be applied to valuable and appropriate automation scenarios in your product's scope?

How could larger scale automations reduce the incidence of certain errors or improve the quality of certain work outputs? What other benefits could result?

What might the user experiences of providing inputs and receiving outputs be like in your sketched functionality concepts? Will workers need to actively monitor your team's envisioned automations? What alerts and cues could guide their observations and awarenesses?

What interaction methods could allow users to locate and override the effects of specific automated steps? How might individuals recognize and recover from certain cases of problematic automation?

What settings and customization functionalities could help ensure that automated processes will operate in accordance with the goals of targeted individuals and organizations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

E5. Visibility into Automation

To help ensure that knowledge workers are not deskilled when they adopt new or revised computing tools, product teams can envision functionality concepts that could provide users with meaningful and useful visibilities into the underlying aspects of certain automated processes.

Examples from three knowledge work domains:

> A financial trader receives a series of automated suggestions in his trading application, based on data that he entered earlier in the day. While reviewing these automated suggestions, he can see the reasons why the application has recommended each potential trade and then make his decisions based on a wider variety of criteria that simply cannot be automated **(see illustration on next page)**.

> A scientist watches as her analysis application pulls from a number of online databases to construct a visualization. The tool color codes content based on its source, highlighting anywhere conflicting information is available from different databases so that she can make decisions about which content to use.

> An architect reviews a log of actions taken by the so called "materials manager" in her building modeling application. She wants to see if she agrees with the "decisions" it made while updating a certain attribute across the entirety of a large building model.

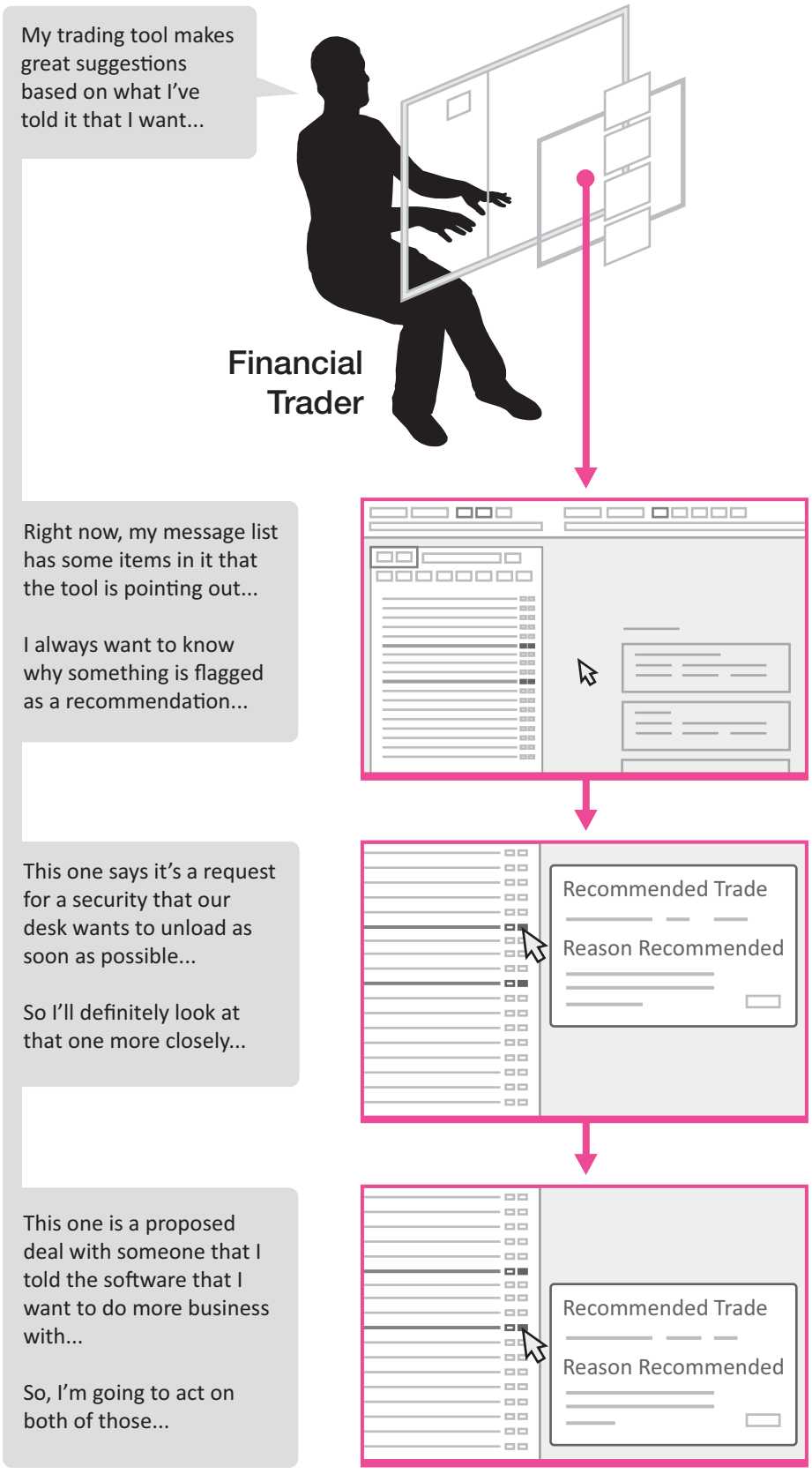
Onscreen user interfaces inherently "hide" many of a tool's inner workings. Sometimes this opaqueness is useful; other times it can deskill. Outside of highly standardized processes (A4, C6), valued technologies in knowledge work may not function as "black boxes" that obscure everything that occurs between the receipt of inputs and the delivery of outputs (G7, J3, L1).

To preserve workers' skills in specific practices, product teams can provide useful and comprehensible visibility into the details of an interactive application's automated actions. Appropriate views of automated procedures can help workers build accurate conceptual models of a tool's functioning (C1), plan the flow of their work around it, and more effectively evaluate critical incidents (F6).

Product teams can explore the notion of visibility as part of envisioning functionality concepts that automate operations, tasks, or larger activities, keeping in mind that the importance of transparency can escalate at higher levels of this hierarchy (A5). At the level of tasks or larger activities, teams can envision options for workers to monitor relevant information about automated processes in real time (B5, D6, C10) or to review stored logs of automated actions after the fact (H2, H3, I7).

When product teams do not actively consider the potential role of visibility into their automation concepts, resulting applications may leave workers feeling hamstrung and without desirable control (E6). Since even well designed automated routines can encounter problems that require human judgment (A, C9, G3), workers may find that diagnosing and fixing issues in these opaque systems takes significantly more effort than the automation was purported to save in the first place (D2, D3).

See also: C5, C8, E, I6, K, M1, M4


My trading tool makes great suggestions based on what I've told it that I want...
Financial Trader
Right now, my message list has some items in it that the tool is pointing out...
I always want to know why something is flagged as a recommendation...
This one says it's a request for a security that our desk wants to unload as soon as possible...
So I'll definitely look at that one more closely...
Recommended Trade
Reason Recommended
This one is a proposed deal with someone that I told the software that I want to do more business with...
So, I'm going to act on both of those...
Recommended Trade
Reason Recommended

Key *application envisioning* questions:

How much visibility might targeted knowledge workers value when encountering or actively using each of the automated offerings in your team's sketched application concepts? When could such visibility be useful; what might it look like; what meaning could it provide; and how present might it be in workers' experiences?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What automations are currently part of the work practices that your team is striving to mediate? What baseline expectations do targeted workers have for visibility into automation processes?

What larger design and technology trends could influence your team's ideas about how visibility into automation might provide value to targeted individuals and organizations?

How might a lack of automation visibility create deskilling barriers to adoption and long term success for your product?

What information about automated operations could be important in the context of different visibility scenarios?

Which of your team's sketched automation ideas might safely remain a "black box"? At what point could pervasive visibility begin to detract from the offloading value of these functionalities?

What role might certain visibilities play during users' initial testing of your computing tool during their adoption processes?

What value might visibility into smaller automations provide? How might this information link out to appropriate settings and instructional content?

How, specifically, could visibility into larger automated processes provide value in targeted workers' practices? Could it primarily be used for real time monitoring or might it become more of a tool for retrospective investigation?

What specialized representations might your team envision to clearly encapsulate and communicate information about automated processes? How might the outputs of automated processes bear meaningful and traceable "signatures" of their creation?

How might early experiences of transparency help knowledge workers build appropriate conceptual models of automation functionalities?

How could refined automation transparency help workers to recover from any critical incidents and standard error cases? How might your visibility concepts tie into your sketched design responses for error handling and functional histories?

What customization options could allow targeted individuals and organizations to tailor automation visibility to meet their local needs?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

E6. Internal Locus of Control

Knowledge workers may sometimes feel that interactive applications "hijack" their work practices in undesirable and stress inducing ways. Product teams can envision their functionality concepts with the intention of promoting a sense of control and mastery in workers' experiences, even as computing tools usefully perform complex actions on their behalf.

Examples from three knowledge work domains:

> An architect runs a tolerance checking function in her building modeling application to check whether one section of a design meets a specific building code. Where the automated function discovers a potential violation, it gives her the opportunity to ignore the finding based on her own interpretation of the particular code's description **(see illustration on next page)**.

> A scientist likes that the latest version of her analysis application allows her to intervene in real time when she sees that automated algorithms are not producing desired outcomes. In the previous version of the same application, she could not interrupt lengthy analyses to make changes.

> A financial trader turns off the automatic trading function in his trading application, which normally takes care of low value, uncontroversial transactions. Accomplishing these deals manually, when he has time, gives him a better sense of his group's standard business.

Knowledge workers may place a high value on how their computing tools automatically perform certain complex actions (E3, E4). But rather than experiencing these tools as yet more technology that "runs itself," workers may want some measure of control over automations (A4, D2), especially when they can influence the character of entire tasks or larger activities (A5, C8, K2, K4).

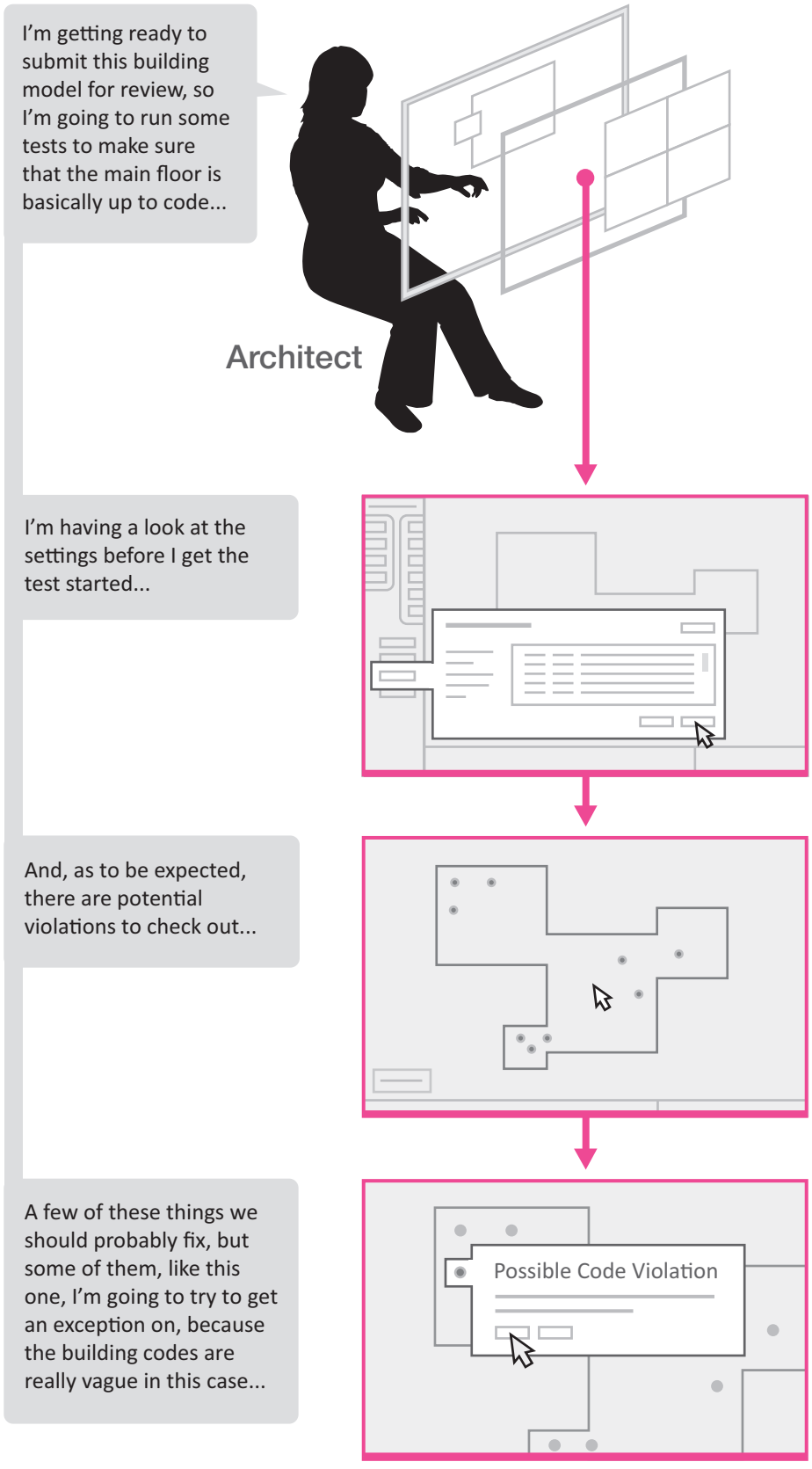
To promote workers' sense that they are at the locus of control, product teams can envision opportunities for users to appropriately contribute their own skills to the initiation, steering, and completion of automated processes (C4, G1).

Over time, workers may build confidence in how an application performs and contributes to their work outcomes (K13, L1), eventually becoming comfortable enough to surrender more complete control of some actions (D4, D7). Product teams can promote these desirable end states by concepting features that could allow workers to transition through such levels of confidence at their own pace.

When product teams do not actively consider how knowledge workers might retain an internal locus of control while using computing tools that powerfully shape their practices, users may find that resulting applications stressfully and inappropriately "make decisions" or "take actions" against their intentions. Workers may believe that they are being deskilled by these computing tools (E5, D3), which can influence their decisions about whether or not to fully adopt them into their own efforts (K).

Conversely, applications can introduce "too much" control, creating unnecessary opportunities for errors (C9, G3) and distracting users from larger goals (D1).

See also: A, C1, E, M1


I'm getting ready to submit this building model for review, so I'm going to run some tests to make sure that the main floor is basically up to code...
Architect
I'm having a look at the settings before I get the test started...
And, as to be expected, there are potential violations to check out...
A few of these things we should probably fix, but some of them, like this one, I'm going to try to get an exception on, because the building codes are really vague in this case...
Possible Code Violation

Key *application envisioning* questions:

What aspects of your team's automation concepts might detract from targeted knowledge workers' sense of agency and skilled accomplishment? How might your computing tool allow workers to have desirable levels of control over the initiation, steering, and completion of automated processes?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What automations are currently part of the work practices that your team is striving to mediate? What do targeted individuals think about their level of control over these technologies?

What problems currently occur due to workers feeling that they are being "controlled" or "reined in" by certain standardized artifacts and computing tools? Could these problems present opportunities for your team's product?

What categorical classes of local needs in targeted organizations might influence workers perceptions of control and augmenting alignment?

What analogies and language might your team use to describe the relationship between user and product that you are striving to create? What implications could this described relationship have on brand?

How might you envision automation functionalities as actionable extensions of workers' skills, rather than distant and self operating replacements for them?

How could thinking about automation as just "another tool" in workers' available repertoires allow your team to sketch more appropriate functionality concepts?

How might a lack of control over certain aspects of your product create deskilling barriers to its adoption and long term success?

How might desired levels of control change over time as users increasingly trust your computing tool?

What settings and options might your team envision to give targeted individuals and organizations meaningful influence over automation functionalities in the context of their local ways of working?

What interactive scenarios and behaviors might provide users with a direct and engaging sense of control over your computing tool's actions?

How much control might be too much control? What constraints could usefully promote reductions in effort, clarified interactive experiences, reduced likelihood of errors, and the confident creation of desired outputs?

What contexts could require automation to be highly standardized, rather than modifiable on a case by case basis at the discretion of individual workers?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F.

Enhancing Information Representation

Valued computing tools can represent information in concise and tailored ways that are well suited to knowledge workers' goals and mental models.

Designing such useful representations requires a deliberate understanding of how people might understand and act upon content.

During *application envisioning*, product teams can critically examine how information is currently represented, looking for opportunities to display important content in enhanced or even transformative ways.

By taking time to generate diverse ideas for their product's information displays, teams can situate new and existing content in comprehensible views that ease navigation burdens and make complex conclusions perceptually clear.

Recorded information, whether inside or outside of an interactive application, exists in specific representational forms. A plain page filled with uniform text is one such form, along with any number of other textual layouts, tables, maps, and graphs. As workers repeatedly create, act with, act on, and communicate through certain representational forms, these standards can become powerful cultural conventions that define and direct shared approaches to thinking within local communities of practice or the entirety of a profession.

Some representational forms can facilitate specific cognitive transformations and work practices better than others. As Herbert Simon wrote, "solving a problem simply means representing it so as to make the solution transparent." Poor representational alignment can interfere with accomplishment, requiring additional thought and action.

Manually creating some representations can require considerable effort — calculating values, laying out a document space, plotting points, filling in areas. By comparison, interactive applications can make generating standard representational forms nearly effortless for their users, opening up opportunities for the rapid exploration of novel perspectives on selected information sets.

This category contains 11 of the 100 *application envisioning* ideas in this book:

F1. Coordinated representational elements

F2. Established genres of information representation

F3. Novel information representations

F4. Support for visualization at different levels

F5. Comparative representations

F6. Instrumental results representations

F7. Highly functional tables

F8. Representational transformations

F9. Simultaneous or sequential use of representations

F10. Symbolic visual languages

F11. Representational codes and context

Product teams can use these ideas to explore a range of concepts for mediating work practices through the dynamic generation and use of different types of information representation. These ideation efforts may help teams to emphasize the importance of existing representational forms or to uncover valuable opportunities for representational innovation. Concepting focused on representation can also allow teams to consider meaningful extensions and interactive transformations of certain information displays, with the goal of further tailoring them toward meaningful ways of thinking and acting.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Defining interaction objects" (B), "Facilitating communication" (J), and "Aiming for aesthetic user experiences" (L) categories.

F1. Coordinated Representational Elements

Elements within and between information representations can have coordinated facets, reducing efforts that would otherwise be needed to usefully bring them into alignment as part of certain operations or larger tasks. Product teams can envision coordinations that could transform effortful mental work into visual judgments and direct manipulations of interrelated external artifacts.

Examples from three knowledge work domains:

> A scientist intuitively transforms a view of clinical data in her analysis application. She gives no consideration to the elegant means by which each transformation stays in synch with other onscreen views, saving her the effort of having to think through and manually navigate these relationships **(see illustration on next page)**.
>
> An architect finds it easy to use printouts from her building modeling application in conjunction with the same building model on her screen. Both the printed and onscreen versions provide the same aligning features, allowing for quick orientation and comparison.
>
> A financial trader views information in his trading application and his market information application at the same time. He changes the date ranges in each tool to the same interval so that he can "eyeball" relationships between the displays.

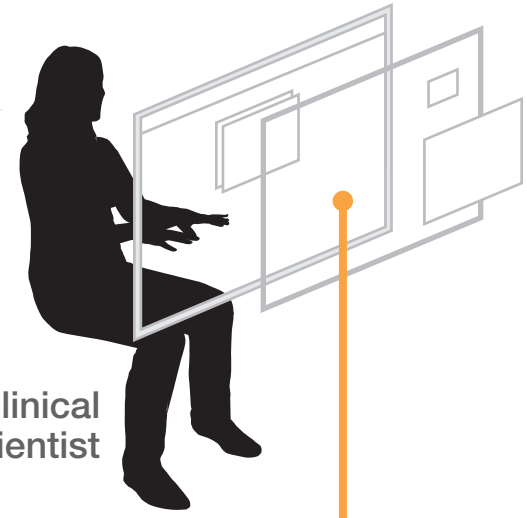
As human beings, we are skilled at making use of and constructing the world around us to enhance our ability to perform complex mental activities (A). Using these skills, knowledge workers often come to understand how different types of information representations "fit" together (B1, F1), providing opportunities to reduce effort (E) and attentional demands (D) in their work.

While people must themselves make coordinations a useful reality in their own practices (A6, A7, A8), product teams can envision how their interactive applications might promote specific threads of meaningful representational connection. These coordinations can transform work by modifying or removing specific mental transformations (D2), changing the nature of, or potentially eliminating, entire operations or larger tasks. In addition to reducing individuals' workloads in valuable ways, clear representational coordinations can also enhance communication and collaboration (C7, G4, J2). After extensive use, workers may become so accustomed to certain facets being coordinated that these relationships may fade from thought, even as valuable linkages are frequently exploited (D4, D7).

When product teams do not actively consider how specific elements of information representations might be coordinated inside and around their sketched application concepts, opportunities to support or positively transform the nature of certain work practices can be lost. When teams overlook coordinations that are currently in use, workers may find resulting tools to be disruptive and frustrating, creating new efforts that were not previously necessary (D3, F11). Teams may also overlook opportunities for novel coordinations with other elements in workers' representational environs (A1), whether internally, within a product's own functionalities, (C4, F9) or externally, with other artifacts, both onscreen and off.

See also: B3, F, G5, I, J6, J7, K5, K6, K13

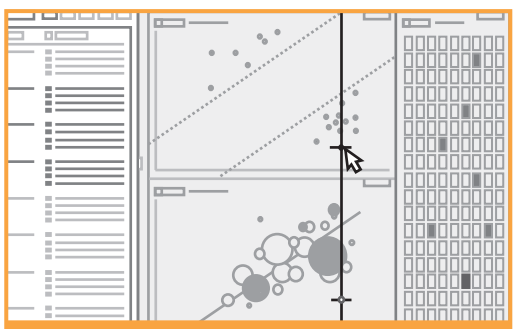
I was just sent a big set of data by a colleague, and I've imported it into my analysis application to look for interesting findings...



Now I'm setting up some connected visualizations before diving in to see what I can find...

And each visualization stays in synch with the others as I make different selections, showing the same highlighted info in each of these views...

And the different views visually line up with each other automatically so I don't even have to think about connecting them together...



Key *application envisioning* questions:

What mental transformations and artifactual alignments do knowledge workers frequently employ in order to manipulate information in goal directed ways? What concepts might your team generate to implicitly coordinate certain meaningfully related elements in your sketched information representations? How might individuals create their own coordinations in the context of your computing tool while performing targeted work practices?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What coordinations within and between information representations, or between certain representations and their larger contexts, do people currently use as part of the work practices that your team is striving to mediate?

What value do current coordinations provide to targeted individuals and organizations? What functional role do these existing alignments play? What problems do they solve?

Which coordinations have become established elements of routine operations and larger tasks? Which are typically more impromptu and variable?

What issues can arise due to representational discoordinations? Could these problems present opportunities for your team's product?

Which existing coordinations will probably not be necessary in the context of your computing tool? Which might become more important?

How might your team incorporate the valuable intents behind existing coordinations into your sketched application concepts? What characteristics of earlier representational forms and interactions could be meaningfully preserved in your product?

What new coordinations might you envision to offload effort and clarify relationships in the context of your sketched functionality offerings?

What interaction and visual design responses could draw attention to and perceptually enhance certain coordinations?

How might your application concepts present "by design" layout consistencies that users could intuitively act within, rather than having to consciously expend effort in order to align certain representational facets?

How might workers create their own representational coordiations by rearranging or reclassifying information within your application concepts?

How could the outputs of your team's computing tool retain useful alignments with onscreen instantiations of the same stored content?

How might representational coordinations play a role in targeted worker's cooperation, collaboration, and communication practices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F2. Established Genres of Information Representation

Knowledge workers reuse established representational formats to create new meaning in a shared interpretive context and to valuably define boundaries for their efforts. Product teams can envision concepts for how these existing genres could be recreated, reinterpreted, and usefully extended in their interactive applications.

Examples from three knowledge work domains:

> A financial trader often says that he knows trade forms better than he knows "his own name." He has used various forms at the different firms where he has worked, though all of them have had the same essential organization and format **(see illustration on next page)**.
>
> An architect uses her building modeling application to generate the types of drawings that are traditionally expected as architectural outputs. While she used to labor over the plans themselves, her team now spends more time focusing on different views of a comprehensive virtual model, from which drawings can be generated.
>
> A scientist views the genetic expression data from a large series of clinical experiments in her analysis application. The data is displayed in a "heat plot," which she is very familiar with after having seen similar visuals in research publications.

Knowledge workers can become highly skilled at making use of information representations that have become standards within their own practices, their organizations, and their larger professions (A, B1). While the evolution of some representational genres can have fairly long historical trajectories, other established formats may have been relatively fixed and unwavering since they first appeared in workers' efforts. The term genre itself implies a certain vagueness in particulars, and named types of information representation may hold diverse variations that workers recognize as having a familial "sameness."

Product teams can envision functionality concepts that usefully incorporate extant representational formats. These established genres can be extended within computing tools to support known variations in workers' goals and approaches (A6, A7, A8), new coordinations with other representations (F1), exploration of potential outcomes (H), integral communication (J1) and collaboration (C7, G4, J4), and long term, organizational memory (E1, I7).

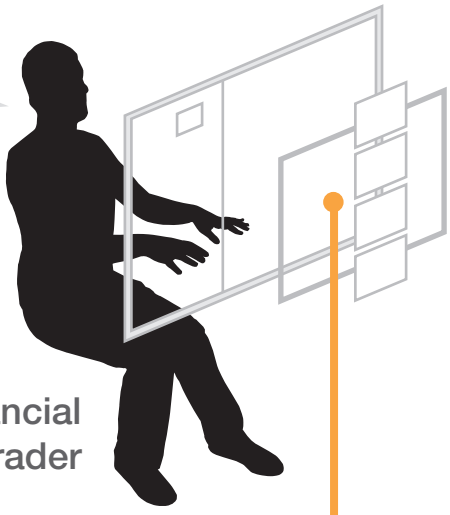
When product teams do not sufficiently consider the potential importance of established genres of information representation in their application concepts, knowledge workers may not recognize resulting offerings as being relevant for their own goals, methods, and roles (K3, L3). Unconsidered re-representation of familiar content may lead to a certain type of deskilling (E6). Without familiar displays of commonly referenced information objects, users may find computing tools to be excessively effortful to learn and use (D2, D3, K2, K6).

Conversely, the tendency for direct, literal translation of established offline genres can prevent product teams from considering how novel onscreen extensions or alternate representations of content (F3, F11) might better meet workers' goals.

See also: B3, E, F, G2, I, J2, L

Everywhere I work, the basics of this business are the same...

You get to know certain screens very well when you look at them over and over every day...



Financial Trader

Trade tickets are a good example...

MINOR DIFFERENCES IN REPRESENTATION

Previous Trade Ticket

Current Trade Ticket

These are the standard ticket forms from my current firm and from the last place where I worked. As you can see, there are only small differences...

And neither of them is really so different from back when these kinds of tickets were paper slips, before the average trader on this desk even used computers...

Key *application envisioning* questions:

What central and long standing representational genres do knowledge workers commonly recreate, derive meaning from, and collaborate around as part of targeted work practices? How might your team incorporate and advance these valued formats within your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How have established genres of representation evolved over time in the tasks and larger activities that your team is striving to mediate?

How, specifically, do people use these known representations? How do defined formats scope and shape workers' efforts?

What do targeted individuals and their organizations think of their standard information designs? What benefits are these genres seen as providing?

Do targeted workers view established formats as essentially immutable or are they open to extending them based on emergent needs and design possibilities?

What errors and misinterpretations can commonly be traced back to the characteristics of established representations? Could these problems present opportunities for your team's product?

How might the onscreen representations of your envisioned interaction objects directly reference any established information artifacts that you have derived them from?

How could preserving existing information designs help workers apply their existing skills and decrease their learning efforts during the adoption of a new product? Where might a change in format provide sufficient value to justify additional effort on the part of users?

Which existing representational genres could be translated into your sketched application concepts fairly directly? Which might require extension or modification in order to effectively make the transition into your computing tool?

How might your team's adaptations of common representational genres provide users with new opportunities for useful coordinations, view transformations, interactive explorations, integral communication, onscreen collaboration, and organizational memory?

How might certain interactions with known displays of meaningful content promote emotional responses that are conducive to attentive, focused thinking?

How might existing genres serve as an inspirational reference for envisioning other, seemingly unrelated functionality concepts?

What impact might the reuse of known representations have on design strategy and brand? What could it mean, in a bigger picture sense, to "conservatively advance" knowledge work in your targeted markets?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F3. Novel Information Representations

Interactive applications can aggregate and display stored data in new ways that are highly useful and meaningful in knowledge work. Within their broader ideas about the advancement of targeted work practices, product teams can identify and explore potential opportunities for new representations of information.

Examples from three knowledge work domains:

> An architect uses a special view in her building modeling application to see what changes have been made to a project over time. The view colors regions of the building model based on how frequently they have been modified. It also provides a timeline slider that allows her to navigate through different versions of the design **(see illustration on next page)**.
>
> A scientist finds that her analysis application includes both representations that are common to her clinical research field and interesting new visualizations that she is not familiar with. After "filling" the new representations with recent data from her lab, she immediately sees their relevance to her work.
>
> A financial trader uses a new interactive graphic in his market information application to view advancing and declining market sectors.

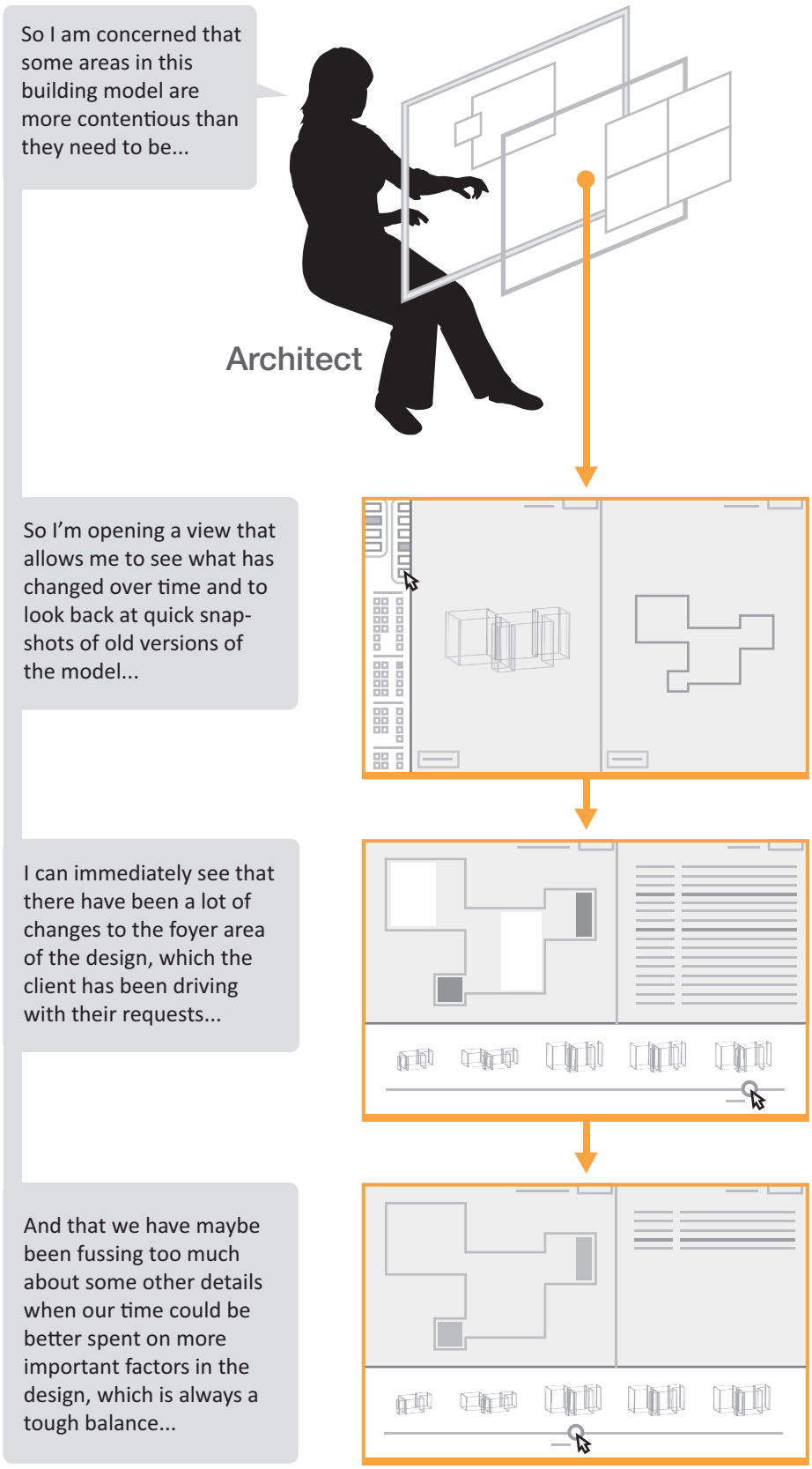
Adopting computing into an activity often means making sense of new forms of visual information. Some established representational needs (F2) may be better met by information formats that are more suitable to onscreen display. Existing representational genres may not scale to adequately present the volumes of content that can arise when mediating work with onscreen applications (F4, I). Beyond these drivers, the introduction of computing power into work can itself open up possibilities for meaningful innovation through the automated generation of complex representational forms (E3, E4).

Product teams can envision innovative representations that are tailored to people's motivations in specific tasks or larger activities. Commonly used representations can be made novel through useful extensions and modifications, potentially for the sake of clearer coordination with other data views (F1). Teams can also introduce novel representations from other domains by making lateral jumps to tangentially related genres based on similarities in purpose, contents, and usage (A, F11).

When product teams do not actively consider the potential role of novel information representations within their concepts for work mediation, opportunities to reduce or desirably transform cognitive effort (E), as well as promote new types of goal oriented understanding, can be lost.

Conversely, established genres of information representation should often be respected as the formats that knowledge workers believe to be the most appropriate for accomplishing their goals (K3, K12). Professionals have often developed extensive skills around the use of existing representations (D7, K6), which may prevent them from seeing value in new approaches (K2). Without a corresponding understanding of their advantages and interpretation, people may perceive novel information displays as being arbitrary and misguided.

See also: C3, F, G2, H, L5, K5, K7, M4


So I am concerned that some areas in this building model are more contentious than they need to be...
Architect
So I'm opening a view that allows me to see what has changed over time and to look back at quick snap-shots of old versions of the model...
I can immediately see that there have been a lot of changes to the foyer area of the design, which the client has been driving with their requests...
And that we have maybe been fussing too much about some other details when our time could be better spent on more important factors in the design, which is always a tough balance...

Key *application envisioning* questions:

How might any deficiencies in current information representations suggest opportunities for representing application content in new ways? What compelling opportunities for representational redesign can be found in your team's sketched functionality concepts? What might these new displays look like, and how could they provide sufficient value to justify knowledge workers learning to use them?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What existing information representations currently lead to breakdowns in the work practices that your team is striving to mediate? Could these problems present opportunities for your product?

Which established representations may not translate well into your application concepts or, more generally, a computer screen?

Which novel work situations within your sketched functionality concepts could be made clearer, less effortful, less prone to error, and otherwise more effective with new representational formats?

How might the aggregation of large volumes of application content suggest opportunities for new representational "containers" that are tailored to meet unaddressed, often higher level, goals?

What larger design and technology trends could influence your ideas about how information in your application concepts could be valuably represented?

What innovative representations of data, whether radically redesigned or entirely novel, might your team sketch as valuable additions to targeted work practices?

Based on your understanding of workers' goals, their current usage of representations, and other factors, what analogous displays from other domains could be applicable to your envisioned computing tool?

How might new forms of representation be usefully and meaningfully coordinated with other information in your application concepts?

How might certain interactions with novel displays promote emotional responses that are conducive to attentive, focused thinking?

What are targeted workers' initial impressions of your team's sketches of novel information representations? How might their perceptions change after more thorough consideration and interaction?

How could your computing tool introduce and frame the value of its novel representations? What instruction and initial scaffolding might be useful?

What impact might the inclusion of new information representations have on design strategy and brand? What could it mean, in a bigger picture sense, to "disruptively advance" knowledge work in your targeted markets?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F4. Support for Visualization at Different Levels

Computing tools can aggregate volumes of content that may be unprecedented within a knowledge work domain. Product teams can envision functionality concepts that could allow workers to visualize aggregated information at different levels of granularity from valuable, goal oriented perspectives.

Examples from three knowledge work domains:

> A scientist navigates through different views of clinical data in her analysis application, narrowing in on areas that show interesting trends. As she selects certain subsets of data, she changes the tool's view to employ specialized visualizations for detailed inspection of smaller result sets **(see illustration on next page)**.

> A financial trader uses his market information application to review recent movements in a range of market sectors. He selects a high volume sector where advances led declines, and the visualization zooms in on the selected area to display its subsectors, along with their individual directionalities.

> An architect is using her building modeling application to review a colleague's project. She views the entire building, rendered as if it actually existed on its large site, then zooms into the front entry space, opting to view only construction notes over unrendered wireframes.

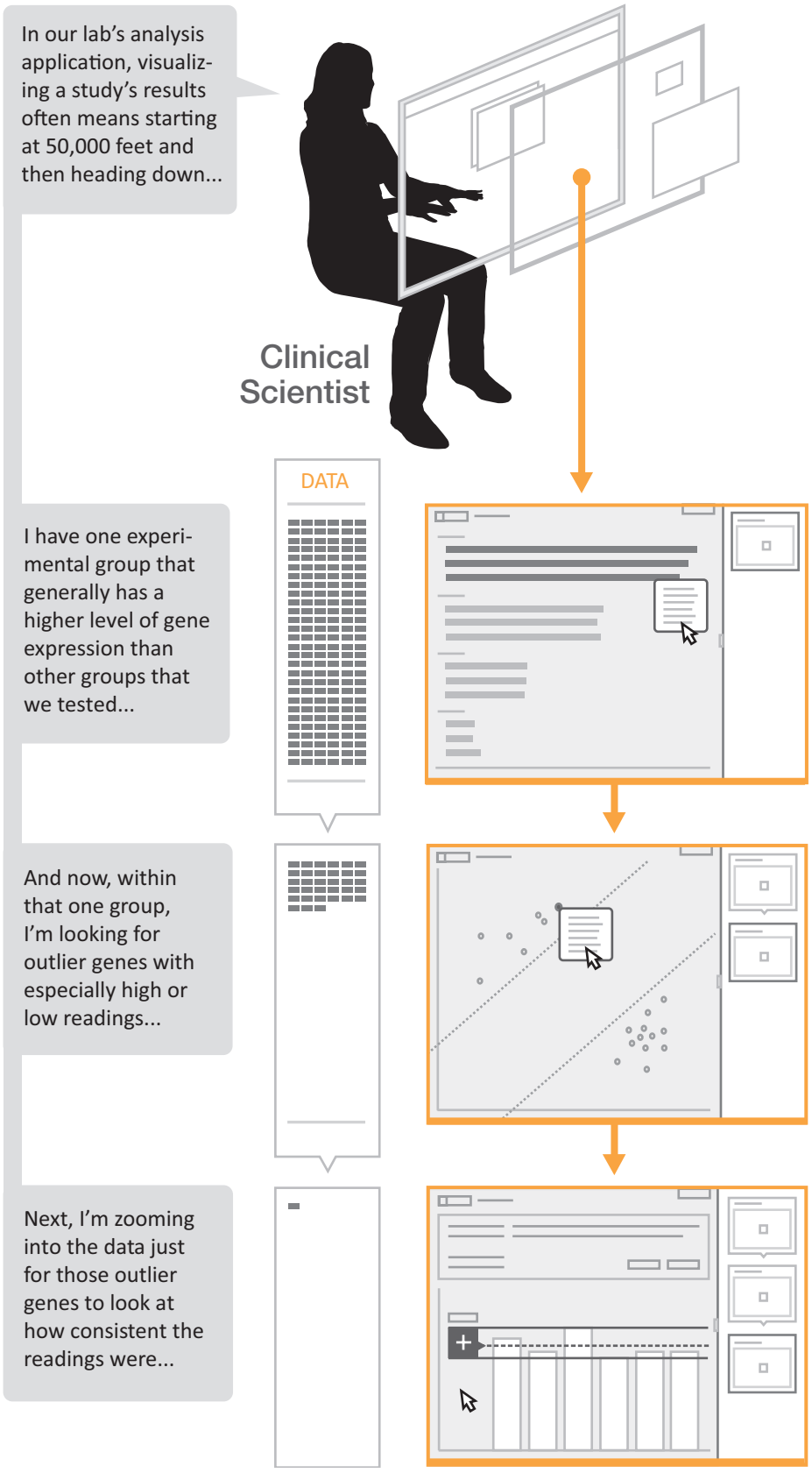
Many established genres of representation in knowledge work are essentially about an individual work item or something that workers think of as a distinct type of artifact (B1, F2). Along side these "ground level" views, some workers may be accustomed to using representations that usefully display content about a number of items simultaneously (I1, I5). Computer generated information representations can take this elevation of scope considerably further, presenting high level "views from the clouds" looking meaningfully down at different aggregations of "ground level" information.

Product teams can envision novel, interconnected series of representations at scaling levels of data concentration. These series may provide compelling support for existing task processes, or present new tools in support of individuals' and organizations' larger goals. Interactions with hierarchical levels of information representation can facilitate exploratory information seeking (A6, G5), promote new types of understanding, and facilitate new approaches to analytical thinking. Novel levels of information aggregation (F3) can be tailored to support relevant problem solving approaches (A) and to provide clear pathways to subsequent actions (C4).

When product teams do not actively consider the potential role of multiple levels of content visualization in their application concepts, opportunities to provide innovative new sources of value can be lost (A9).

Conversely, in some domains, knowledge work revolves around entirely discrete items in clearly articulated processes (A4, C6). In these cases, individuals and organizations may not perceive higher level visualizations as being especially valuable additions to their efforts (D1, D4).

See also: C3, E2, F, G2, H, I, K2, K6, L


In our lab's analysis application, visualizing a study's results often means starting at 50,000 feet and then heading down...
Clinical Scientist
DATA
I have one experimental group that generally has a higher level of gene expression than other groups that we tested...
And now, within that one group, I'm looking for outlier genes with especially high or low readings...
Next, I'm zooming into the data just for those outlier genes to look at how consistent the readings were...

Key *application envisioning* questions:

How might the storage of large volumes of information in your team's application concepts provide opportunities for innovative interactions and insights in targeted knowledge work? What types of information representation could make sense at different levels of content aggregation? How might these scaling perspectives be usefully interlinked in support of certain analytical goals?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Where might volumes of stored data overload the representations that people currently use in the work practices that your team is striving to mediate?

Why might targeted individuals and organizations want to visualize information at different levels of aggregation? What problems could scaling levels of information representation solve?

How might new levels of information display meet unaddressed goals in targeted tasks or larger activities? What aspects of these new displays could offload effort or enhance certain lines of analytical thought and explorative sense making?

Based on your team's understanding of workers' goals, their current usage of representations, and other factors, what analogous displays from other domains could be applicable to your envisioned directions for scaling data visualizations?

What larger design and technology trends could influence your ideas about how information in your application concepts could be valuably represented and navigated at different levels of concentration?

What novel concepts might your team sketch for higher volume information representations that are tailored to targeted knowledge work goals?

How could workers usefully navigate through connections between different levels or represented information? What meaningful frameworks and interactive transitions might your team envision to clarify the relationships between representational strata?

How could your computing tool introduce and frame the value of new systems of interrelated displays? What instruction and initial scaffolding might be useful while individuals are learning to use these new representations?

How might your team's ideas about supporting visualization at different levels relate to your other design responses for supporting work in the context of volumes of information?

What impact might the inclusion of new visualization approaches have on design strategy and brand? What could it mean, in a bigger picture sense, to "disruptively advance" knowledge work in your targeted markets?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F5. Comparative Representations

Knowledge work can involve standard comparisons, based on known and meaningful criteria, between work artifacts. Product teams can envision functionality concepts that automate certain comparisons between interaction objects and display resulting outcomes in representations that highlight any distinctions that are pertinent to workers' goals.

Examples from three knowledge work domains:

> A financial trader chooses an option in his trading application to compare all available offers for a particular security. A special visualization highlights the differences between six offers that are currently available, visually emphasizing the most important characteristics and the magnitude of their discrepancies **(see illustration on next page)**.
>
> A scientist selects two categories of clinical data in her analysis application so that she can view a summary of differences between them. The application presents her with a visualization that graphically illustrates key distinctions in the data across several variables.
>
> An architect uses a feature in her building modeling application to compare two saved versions of a particular floor plan in a hospital proposal. The resulting view is a composite that assigns each version a color and removes all features that are precisely shared. Only the differences remain salient, in bright colors that call out which version of the model is the source of each discrepancy.

Knowledge workers often make comparisons manually, without specialized representations for the task, by placing multiple printouts (J7), onscreen windows, or other artifacts within their visual field and scanning pertinent features (B1, G5). In some cases, individuals and organizations may define standard information displays that crystallize and bound certain comparative tasks (F2).

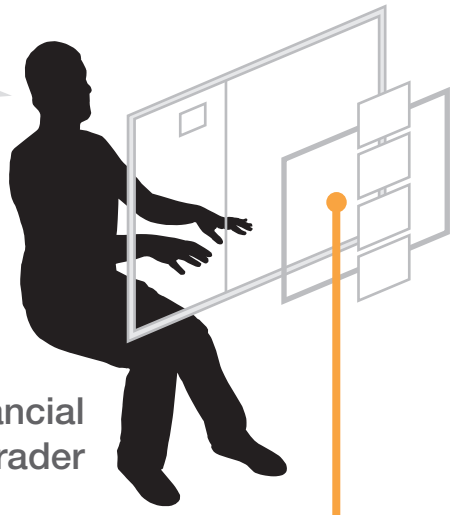
Interactive applications can excel at automating comparisons (E3, E4) and displaying resulting outputs in representational formats that call out meaningful distinctions in informative ways (A).

To envision displays that make comparative conclusions clear (C4, G1, F10), product teams can explore concepts for adapting established representations already used within targeted work practices. Teams can also ideate around workers' concrete comparison needs in order to generate concepts for more novel representations (F3, K6). Depending on the bases of comparison (B6) and how standard individuals' decision making criteria are (A4, A8, C8, F6), effective comparative representations may be categorically different from how the objects under comparison are typically displayed (F8).

When product teams do not actively consider the potential role of comparative representations in their application concepts, opportunities to improve certain types of decision making and reduce or eliminate tedious, repetitive operations can be lost. People may find the exacting nature of manually comparing application content to be excessively effortful (D2, D3, K2) and error prone (C9, G3), increasing their short term memory burdens (E2) and reducing time spent on their higher order goals.

See also: B3, E, F, G6, I, J6, K4, K5, L

I have to quickly fill this order...



The search results show that we have four different sources for the security that I need...

All four seem relatively similar, so I'm going to use the comparison view...

I love the way this screen calls out differences in the info that I care about, including some more complex analyitics...

And I'm removing sources that don't look right...

So, it looks like it's a toss up between the first two...

And the tool has put them first because its rules generally know what I look for when making these decisions...

Key *application envisioning* questions:

What comparisons do targeted knowledge workers frequently make in the work practices that your team is striving to mediate? What specialized information representations could allow workers to accomplish valuable comparisons by quickly interpreting emphasized distinctions between selected interaction objects?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What types of information artifacts do targeted individuals frequently compare?

What are some common bases of comparison? Which can be especially important in targeted operations, tasks, and larger activities?

Which comparisons are currently accomplished manually, by workers' placing multiple information representations in their visual fields, switching back and forth between screens, or other ad hoc methods?

What comparative representations do workers currently use in their established practices? What value do these formats provide?

What memory efforts and cognitive load are involved in particular types of comparisons? Are these acts relatively easy to accomplish, or do they present burdens that could be valuably reduced by your team's product?

Where might automated comparisons of application content provide valuable new support for analytical judgments and explorative sense making in targeted work practices?

What larger design and technology trends could influence your team's ideas about how information in your application concepts could be comparatively displayed?

What improvements and extensions might you envision for existing comparative representations as part of incorporating them into your application concepts?

What novel comparative displays might your team sketch, based on your understanding of workers' goals and current practices?

How could your computing tool introduce and frame the value of novel comparative representations? What instruction and initial scaffolding might be useful while individuals are learning to use these new displays?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F6. Instrumental Results Representations

For knowledge work processes where the desired user experience is highly automated, "push button" simplicity, product teams can envision distilled representations of resulting information outputs that could facilitate rapid judgments within targeted work practices.

Examples from three knowledge work domains:

> A scientist uses her analysis application to test whether any of the subjects in her clinical study, based on a subset of their uploaded genetic information, have a predisposition for certain well characterized conditions. She is surprised by how easy this test is to run and how concisely the results are displayed **(see illustration on next page)**.

> An architect runs a test in her building modeling application to simulate how light will pass through windows into a building's interior over the course of a day. Almost immediately, the tool highlights areas of the model's floor plan that do not receive a threshold value of natural light.

> A financial trader sees a glitch in his trading application and chooses to "test the connection" between his tool and an information vendor. The test automatically progresses through a series of checks, then displays a conclusive "passing" result.

As certain processes become standardized and increasingly automated (E3, E4) in knowledge work, individuals may begin to expect the rationality of what Davis Baird has called "instrumental objectivity." In these user experiences, which are common in mature consumer product genres, certain tasks or even entire activities (A5) that were previously effortful and required specialized skills become streamlined (A4) to a few simple input (B1, B3) and output steps (L1).

As a side effect of automation in an "instrumental objectivity" style, workers' conceptual models of underlying processes may become uncritical, limited, or even distorted (C1, D4, K7). These losses in understanding may be viewed as a positive impact, as an acceptable trend, or as a clear problem by certain individuals, communities of practice, organizations, and professions at large.

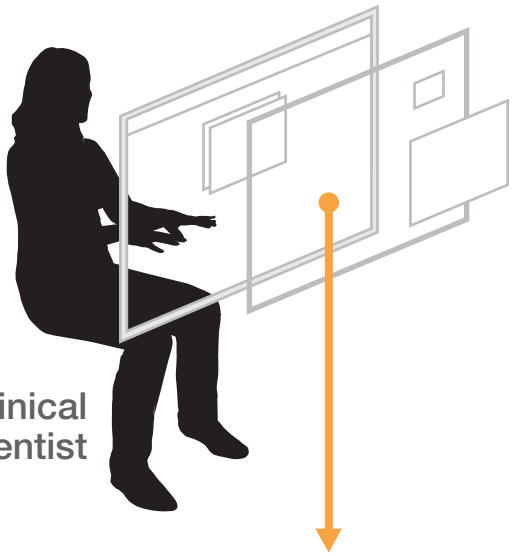
With these potential effects in mind, product teams can envision how automated scenarios in their sketched functionality concepts could result in information representations that provide users the "answers" that they are seeking, embedded within relevant context. These rationalized outputs can also clarify potential next steps (B5, B6) by presenting pathway options within the larger narrative of workers' activities (C4, G1).

When product teams do not actively consider the potential role of instrumental results representations in their application concepts, opportunities to create meaningful innovations in summarized information display can be lost. When workers expect these highly concise and directive outputs, anything else may seem unnecessarily complicated (D2, D3) and difficult to learn (K2, K6).

Conversely, in some cases, distilled representations of automation results can inappropriately oversimplify work outcomes in misleading ways, especially when functionality to view more detailed, underlying information is not provided (E5, F4, G3, K5).

See also: A, C9, D6, E, F, I, J, K4, K12

I have a large set of clinical data, and I want to run some basic tests on it to see if there are any known, major genetic abnormalities in the subjects...



So I've selected the data from the new subjects in my analysis application, and I'm choosing the range of testable abnormalities that I want the tool to look for...

And a few seconds later, when the results have come back, it gives me a quick summary of how many abnormalities were found...

I can then scroll down through the results to see the genetic conditions for each subject, organized by statistical confidence and the severity of potential health impacts...

Key *application envisioning* questions:

Which of the knowledge work tasks or larger activities that your team is striving to mediate could be valuably supported by automations that result in easy to interpret, "instrumental" outputs? How might these results be distilled into meaningful representations of clearly actionable information?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What do targeted individuals and organizations think about the simplification of certain work practices into instrumental inputs and outputs?

What types of instrumental results representations do knowledge workers currently use?

Which existing tasks have conventionally become so automated that even experienced workers have nearly forgotten how they could be accomplished without their current technological support?

What are targeted workers' expectations about "push button simplicity" in the activity contexts that your team is targeting?

Where in your team's application concepts might you valuably cultivate this sort of highly trusted offloading in new scenarios? What standard and tedious work practices could be automated to an extent where people may not need to monitor or comprehend their inner workings?

Where might this kind of simplicity become an unwanted barrier to workers being able to use their own analytical, sense making, and procedural skills in fine grained ways?

What larger design and technology trends could influence your team's ideas about how output content in your application concepts could be instrumentally represented?

What analogous representational conventions might you reference, or apply directly as patterns, to your envisioned output displays? How might these analogies enhance users' intuitive understanding of certain readouts?

What standard output states might your sketched automation concepts result in? How could these states drive appropriate variations in representational responses, as well as the clear presentation of relevant pathways for subsequent actions?

How might instrumental results displays surface ambiguities and errors in the execution of rule based processing? How could these representations reference your larger standards for error prevention and handling?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F7. Highly Functional Tables

Tabular representations are pervasive in knowledge work. Based on an understanding of how various tables in an application concept might be used, product teams can envision functionalities to powerfully transform and extend gridded content to meet certain goals and analytical conditions.

Examples from three knowledge work domains:

> A financial trader typically has several of his trading application's tables open on his screens at the same time, displaying available assets, offers, booked deals, trade balances, and other meaningful categories of information. While making trading decisions, he often searches and manipulates these tabular views to locate and examine specific information **(see illustration on next page)**.

> An architect uses tables in her building modeling application as alternate or complementary views to the 3D building form in a project's file. She finds that these tables are often useful when she is looking for named objects in a design that she cannot remember the location of spatially.

> A scientist frequently uses tables in her analysis application in conjunction with graphical visualizations of clinical data. When she has spotted an interesting trend in an interactive graph, the complementary tables contain the detailed information that she needs in order to make sense of specific results from a variety of exacting perspectives.

Tables, one of the oldest forms of information representation, are a crucial focus in many knowledge work domains. Within interactive applications, tables can become highly dynamic and transformable displays of content (E3, F8, I6).

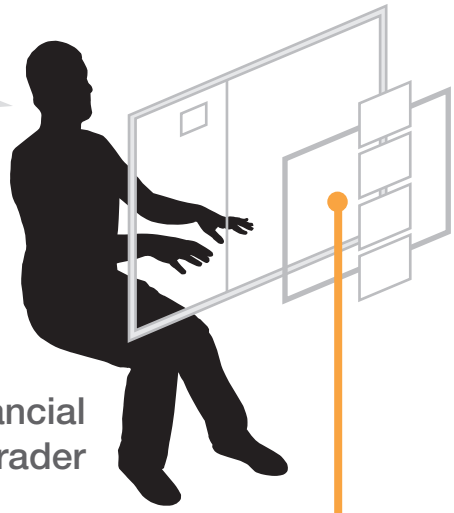
Product teams can envision systemic approaches for table functionalities across their sketched ideas for work mediation. For example, teams can define categories of tables within their application concepts and then consider the level of functional complexity needed for each category. Classification of tables can be driven by the volume of data that they will likely contain (I) and the specifics of how they are indented to be used in workers' practices (A). Useful functional responses for tabular views can include comprehensive search (I2), reorganization and customization (I1, C8), filtering and sorting (I3), search programming (K11), printing (J7), and direct data entry interactions (B2).

When knowledge workers are accustomed to using powerful table functions in other applications, such as spreadsheet products, they may develop high expectations of gridded displays in their other computing tools. In some cases, extensive table requirements can be sufficiently met through lightweight interoperability with other products (K8) or the ability to export selected sets of tabular data (K9).

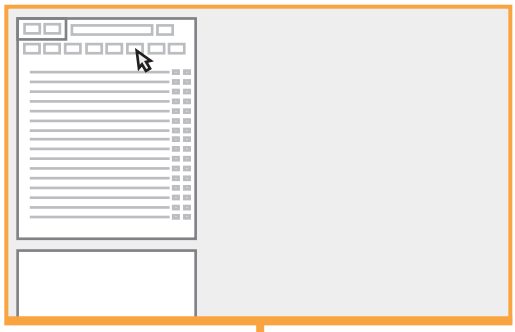
When product teams do not actively consider the appropriate level of functionality for various tables within their application concepts, resulting products may present knowledge workers with inconsistent and underdeveloped options relative to their needs and expectations. When users have to extensively scan through rows and columns (D2, D3), they may overlook important information and incorporate less relevant content into their work outcomes (G3, K5, L1).

See also: B5, B6, C3, C4, C8, G2, G5, F, H, J4, J5

I nearly always have a list of trading messages that I need to go through...



When it's busy and my message list gets really long, I am constantly changing the ordering to see what's going on...



As I look at each message, I'm checking the related tables below to make sure that I know the bigger situation around it...

And as I think of questions about a potential deal, I can add on to or change these tables to quickly find the answers that I need...

Key *application envisioning* questions:

How might your team categorize tables across your sketched functionality concepts based on the volume of their potential contents and their associated goals in targeted knowledge work? What types of interactive offerings could be usefully and consistently applied to different categories of tables? How might other representations coordinate with gridded views as part of certain operations and larger tasks?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How are tables currently used in the work practices that your team is striving to mediate? Might other types of information representation support these goals more effectively?

What expectations for table functionalities have targeted individuals developed from using other interactive applications? What standard or unusual table options do they value in their computing tools?

What opportunities for tabular representation are inherent in your team's sketched functionality concepts? How might these tables be supplemented with alternate views of the same application content?

What design options might you envision with the goal of making your product's tables more than just flat lists of data? How might the interactive and contextual appearance of certain visual cues call attention to important line items?

What functional options could be valuable for different categories of tables within your application concepts? How might certain options support workers' information seeking, content organizing, and sense making goals?

Which grids in your team's envisioned functional areas could become a frequent focus of workers' attentions and activities? How might more extensive functionality, such as specialized searching, filtering, and sorting options, provide value in these central tables?

Which lists in your sketched application directions, by contrast, could benefit from the simplicity of very limited functionality?

Where might interaction requirements be extensive enough to suggest that workers' practices could be better supported through clear and direct transfer of content to a supplementary, feature rich computing tool, such as a spreadsheet?

How might your team's ideas about highly functional tables relate to your other design responses for supporting work in the context of volumes of information?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F8. Representational Transformations

Knowledge workers may use a single information representation as part of accomplishing very different work practices. To support differing needs from a single information display, product teams can envision functionality concepts that could allow workers to meaningfully tailor how a representation classifies and presents selected content.

Examples from three knowledge work domains:

> An architect likes that she can change the contents of views in her building modeling application based on what her current goals happen to be. For example, she can view the 3D model as a full color, rendered building form, or as transparent wire frame geometry. She also has options to visually highlight different features of a building's design that have certain identities tagged to them, such as ventilation or lighting systems **(see illustration on next page)**.

> A financial trader wants to close out the day by increasing his trading volume with some of his best business relationships. He chooses options in the "incoming offers" table in his trading application that will reduce the extremely long list of potential deals to a visually categorized set of promising proposals made by his preferred firms.

> A scientist is looking for outlier data in the results of a clinical study. She changes a color coding scheme in her analysis application so that only data points with very high or very low values are highlighted in a dynamic visualization.

Knowledge workers may adopt valued information representations into a variety of different practices (K), establishing or improvising (A6, G5) approaches to using a display in the context of diverse motivations and constraints. Workers may have gone so far as to develop small variants of often used representations in order to advance their applicability in particular tasks or larger activities (A5, F, D4).

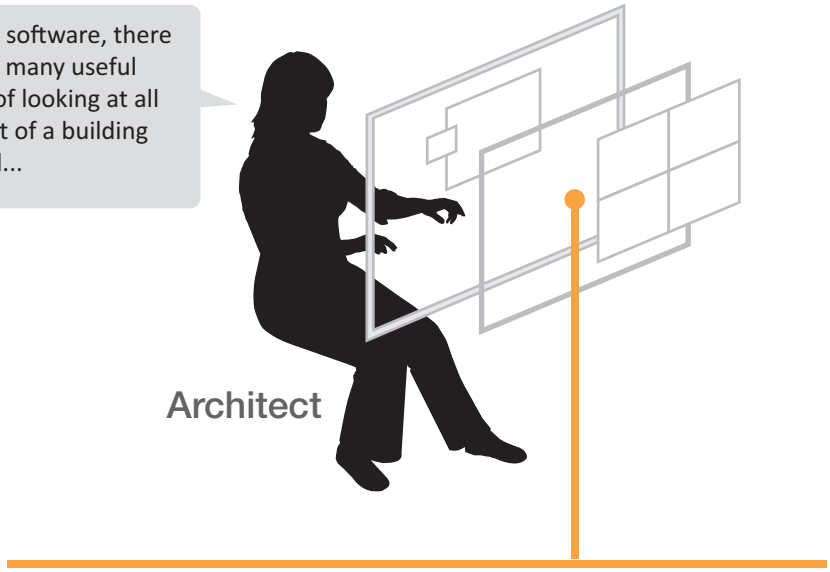
Product teams can envision functionality concepts that could allow workers to visually reclassify and reformat displays of application content in order to better highlight certain features in information sets (B6, F3, J4). By taking advantage of our innate human ability to recognize visual patterns (F7), these transformations can significantly reduce the effort that workers need to expend (E3, E4) in order to accomplish specific information seeking (I2, I3) and sense making goals.

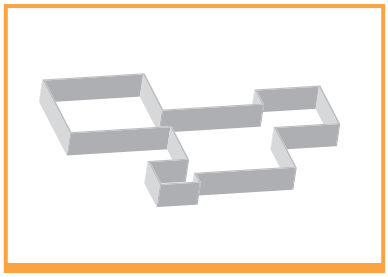
It is worth noting that supporting certain transformations of information displays does not mean removing meaningful defaults for them (C4). Teams can balance notions of representational flexibility (A9, C8, E6, M4) with requirements for initial learnability and ongoing usability (D7, K) in key scenarios.

When product teams do not actively consider how sketched representations in their application concepts could be appropriately transformed to meet workers' varying orientations, opportunities to reduce effort (D2, D3) and promote new sources of clarity can be lost. Resulting computing tools may not sufficiently support existing local practices (A8). Perhaps most importantly, users may not uncover and incorporate valuable insights into their work outputs (L1).

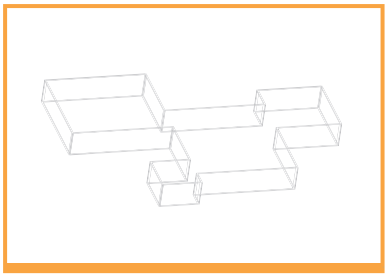
See also: A, C5, F, H, I, L

In this software, there are so many useful ways of looking at all or part of a building model...

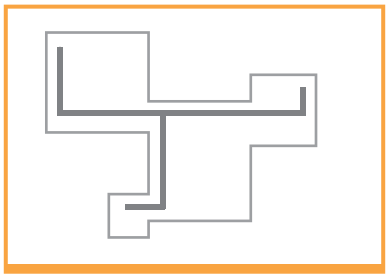




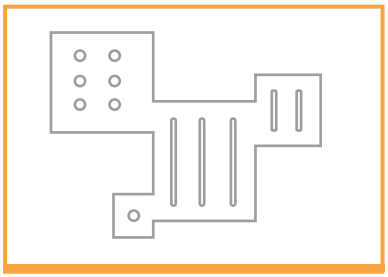
Rendered building of one floor



Wireframe geometry of one floor



Ventilation systems within one floor



Lighting elements within one floor

Any one view can be transformed to show or hide all sorts of different data...

So I turn on what I need based on what I'm trying to do...

Also, the application is surprisingly smart about suggesting different visual transformations based on what it gathers about my current needs...

Key *application envisioning* questions:

Which of your team's sketched information representations could be used in multiple work practices — especially in distinct information seeking and sense making efforts? What functional options might allow targeted knowledge workers to visually transform these representations in support of certain characteristic or emergent needs?

More specific questions for product teams to consider while envisioning applications for knowledge work:

In which separate tasks or larger activities do targeted individuals use the same information representations? How do these usages vary?

How might differing uses of a single representation suggest opportunities for valuably transforming it to meet important scenarios in targeted work?

What larger design and technology trends could influence your team's ideas about how information displays in your application concepts could be manipulated around diverse goals and constraints?

What visual changes might your team envision to usefully highlight various types of meaningful differences within a single display?

How might interactive transitions between your sketched view transformations promote certain types of clarity and meaning? How could these navigation actions draw perceptual linkages that may enhance coordinations in users' efforts?

At what point might a transformed information representation become an entirely different view of application content, rather than a different take on the same type of display?

What demographic and localization requirements might your team consider while envisioning representational transformations?

How might certain goal driven, interactive display changes promote emotional responses that are conducive to attentive, focused thinking?

When could transformed views become individuals' preferred perspectives on application content? What options could usefully facilitate customizable defaults?

How might transformation of shared representations lead to breakdowns in common ground for communication and collaboration?

How could your computing tool introduce and frame the value of certain view transformations? What instruction and initial scaffolding might be useful while individuals are learning to use these new display methods?

How might your team's ideas about representational transformations relate to your other design responses for supporting work in the context of volumes of information?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F9. Simultaneous or Sequential Use of Representations

Knowledge workers may use more than one information representation, of the same or different content, to accomplish certain operations or larger tasks. To support workers' abilities to meaningfully act from the context of different data perspectives, product teams can envision concepts that present certain displays in parallel or allow for rapid switching between related views.

Examples from three knowledge work domains:

> A financial trader selects a single offer from a table in his trading application, then opts to view a large display of historical graphs related to the offer. Since the proposed deal does not look advantageous, he moves on. To browse potential deals more effectively, he sets up his screens to view the application's large table of offers and some related graphs at the same time so that he can quickly investigate historical data for each proposal that he selects **(see illustration on next page)**.

> An architect is tasked with checking the lighting modifications that a consultant just completed. She supplements the main 3D visualization in her building modeling application with a floor plan that highlights all lighting elements, as well as a table that lists all of the lighting fixtures in the current design and their linked product specifications.

> A scientist switches between different visualizations in her analysis application, some of which she rarely uses, hoping to unexpectedly discover some insight about a small but interesting collection of samples within a large clinical data set.

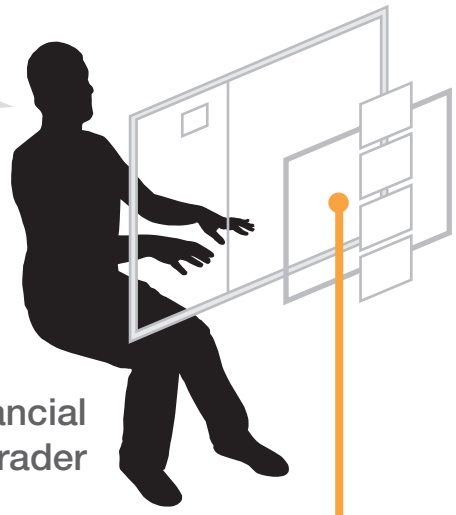
Some knowledge work tasks or larger activities can require, or at least benefit from, the use of multiple, coordinated representations (A5, F1). Workers may act on and through a number of different types of information at the same time, each in their own tailored format. Additionally, people may find value in viewing multiple perspectives on the same content, or the same type of content, potentially at different levels of detail (F4).

Interactive applications can facilitate representational juxtapositions that workers currently find valuable while at the same time opening up opportunities to quickly view information from more orientations. Product teams can envision application concepts that could dynamically display multiple views of stored content in meaningful configurations (E3, E4). They can also sketch default arrangements and sequences of information displays that could simplify common scenarios in workers' practices (A4, K6). Where additional flexibility may be useful or required (A9, F8), teams can consider customization options for tailoring onscreen perspectives in support of specific motivations and constraints (A8, C8, E6, M4).

When product teams do not actively consider how knowledge workers might simultaneously view and transition through multiple information representations, opportunities to promote valuable coordinations, interaction efficiencies (C4, G2), and insights across views can be lost. Important representations may become isolated in limiting and fixed frames, potentially leading to user frustration (D2, D3, D4), increased memory burdens (E1, E2), and excess printing (J7).

See also: A, C, F, G5, H, I, J2

People talk about getting overloaded with too much information, but I like to have the option of seeing a lot of different kinds of data at the same time...



Financial Trader

I'm turning on some graphs here to see what's going on with this potential deal...

No dice here...

And I've got a lot of other messages to go through, so I'm going to close down this middle column of details for the time being...

So now I can just look at specific messages and their graphed data, which feels faster sometimes...

Key *application envisioning* questions:

How might close onscreen relationships between coordinated displays of information provide value in the knowledge work practices that your team is striving to mediate? What sequential or simultaneous arrangements of content in your application concepts could allow targeted workers to more easily see key relationships or interact through them more directly?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What types of information artifacts are currently used in conjunction with each other in targeted tasks and larger activities?

When and how do workers use coordinated aspects of these artifacts in parallel or in sequences? What value do these connectivities provide?

How might your team use these understandings to envision useful possibilities for relationships between views in your sketched application directions?

What larger design and technology trends could influence your ideas about how information representations could be displayed in conjunction with one another in your computing tool?

How might your team's ideas for novel data views be used in conjunction with more established representations of domain content?

How might related displays be meaningfully sequenced? What dynamic pathways could link and bridge higher level views of application content with known and established lower level views?

How might interactive transitions between your sketched views promote certain types of clarity and meaning? How could these navigation actions draw perceptual linkages that may enhance coordinations in users' efforts?

What implications might your ideas about relating various information representations have on the overarching frameworks of your application concepts?

At what point might targeted individuals perceive new display functionalities as being too complex for their own work practices? How might your application concepts retain a refined clarity and appropriate levels of simplicity?

How could your computing tool introduce and frame the value of multiple views of application content? What instruction and initial scaffolding might be useful while individuals are learning to use these new display possibilities?

How might your team's ideas for simultaneous and sequential representations relate to your other design responses for supporting work in the context of volumes of information?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F10. Symbolic Visual Languages

Symbology can be a central component of interactive applications, adding clarity and emotive style to representations of onscreen objects, interactive options, information categories, or messaging content. Product teams can envision symbolic approaches for their application concepts that meaningfully advance and extend known visual languages.

Examples from three knowledge work domains:

> A scientist navigates a visualization in her analysis application that displays the results of a clinical study as they relate to known functional pathways of human biology. Different elements of these biological pathways are represented as specialized, interconnected, iconic symbols, which become highlighted based on their relationship to the clinical data set that she is investigating **(see illustration on next page)**.
>
> An architect appreciates that her new building modeling application incorporates a large selection of symbols that are conventionally used in architectural drawings, along with some potentially useful new ones.
>
> A financial trader has customized certain tables in his trading application to include categorical icons based on a row's contents. During stressful times in the trading day, he values how these small cues allow him to quickly interpret incoming information without scanning many specifics.

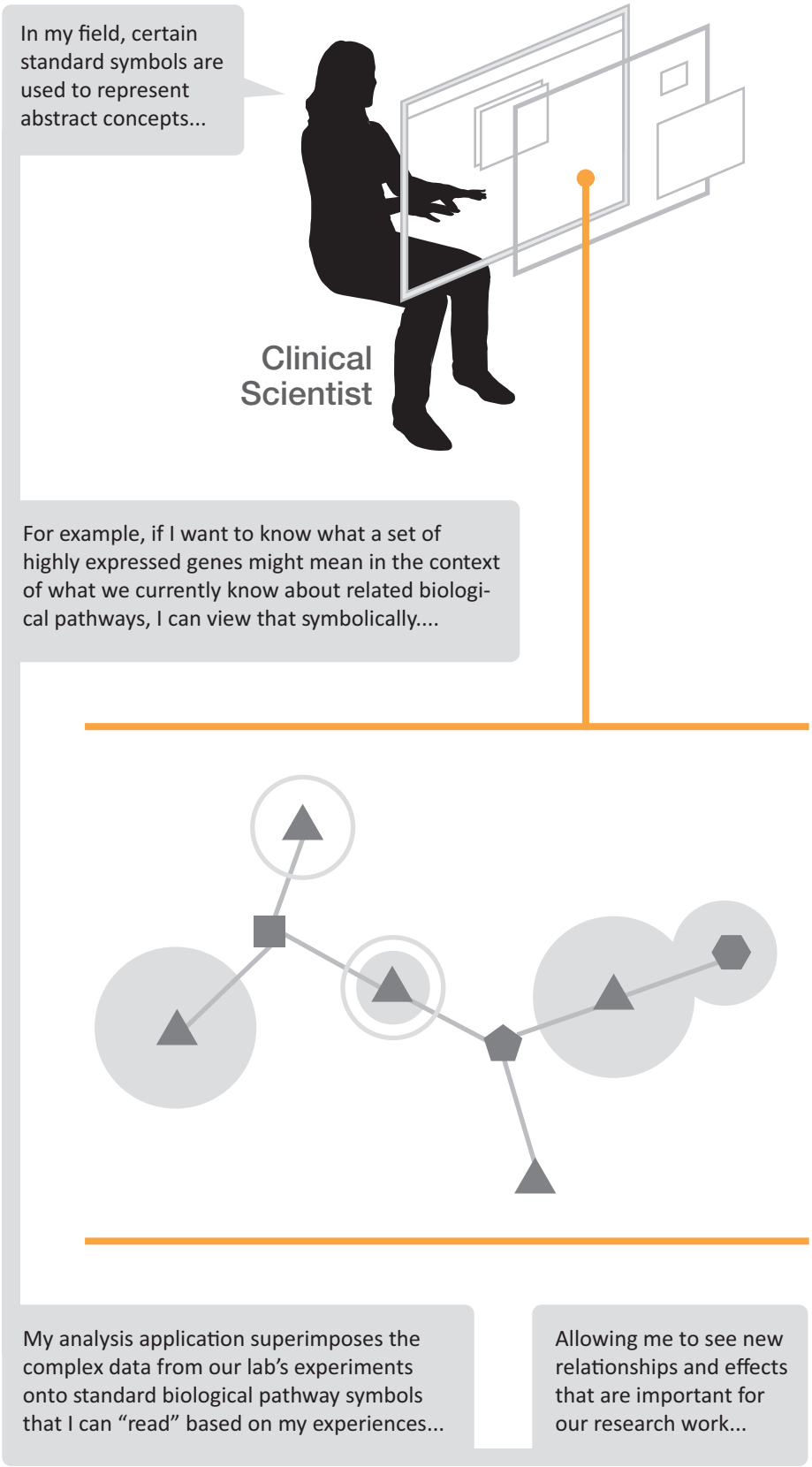
Symbolic visual languages can range from color coding to literal iconography. These languages can serve many purposes, both within an application's overall framework (C) and embedded within its information representations. Iconic symbols are often used to represent different varieties of interaction objects, as well as entry points to various interactive pathways, as in a conventional toolbar or menu of options (C3, C4). Non textual, symbolic cues can provide value as indicators of category (B5, B6) or as a method of communicating some types of messaging and instructional content (K2, K7, C9, G3).

Product teams can envision approaches to symbolic language that are built on both contemporary conventions in application design (L2) and specialized symbolic systems that have evolved within targeted knowledge work domains (A1). These established forms (F2) can be incorporated into products essentially as is (K3) or can serve as a foundation for further concepting (L3, L4) and styling.

Since shared interpretation of abstracted symbols is often an issue, especially across cultures (K1), supplemental information about symbolic cues may be necessary or at least recommendable (F11, K5). This supporting content may be persistently visible or made available upon demand, depending on a variety of factors, including predicted frequency of use.

When product teams do not actively consider the potential role of symbolic visual languages in their application concepts, opportunities to effectively communicate certain types of information can be lost. Without the reductive visual power of symbolic representation, knowledge workers may find some displays in resulting applications to be difficult to quickly assess and somehow uncompelling (D4).

See also: A, C8, D7, F, L


In my field, certain standard symbols are used to represent abstract concepts...
Clinical Scientist
For example, if I want to know what a set of highly expressed genes might mean in the context of what we currently know about related biological pathways, I can view that symbolically....
My analysis application superimposes the complex data from our lab's experiments onto standard biological pathway symbols that I can "read" based on my experiences...
Allowing me to see new relationships and effects that are important for our research work...

Key *application envisioning* questions:

What symbolic conventions are currently used in the knowledge work practices that your team is striving to mediate? While referencing these existing languages and the conventional iconographies of interactive applications, what new concepts might your team envision to symbolically communicate information and affordances in your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How are symbolic visual languages currently being used in the larger professions and industries that your team is targeting?

What symbologies are targeted individuals familiar with from interactions with other products and other life experiences?

How might your team use these known conventions as a starting point to envision meaningful and branded symbolic visual languages in your application concepts?

What larger design trends and advanced analogies to other domains could influence your ideas about how symbology could take shape in your computing tool?

Where could symbolic information representation be an effective means of design communication in your team's functionality concepts and information representations? What clarifying and enhancing value could symbols provide in various situations?

What information rich and real estate constrained functional areas could benefit from iconic communication of application content? How might perceptually salient cues call out important information?

Where could symbolic representations improve the interpretation of instructions and textual descriptions?

How might your requirements for learnability in various functional areas influence your decisions about where to apply meaningful symbolic cues?

How might your team's design responses for symbolic languages relate to your ideas about illustrative content?

How might your team envision the symbolic communication in your application concepts as an overall system that is a complementary element of a larger aesthetic direction and brand?

What are the demographics in your targeted markets? How might your concepts for symbolic content be interpreted by different cultural audiences?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

F11. Representational Codes and Context

Information representations may require supporting content in order to be interpreted correctly by knowledge workers. Product teams can envision how different representational forms in their sketched application concepts could be clarified with useful labels and keys, as well as descriptions of current data scope.

Examples from three knowledge work domains:

> An architect sometimes gets confused while rapidly navigating the virtual space of her building modeling application. She often turns on the application's scale indictor and a small overview map of the building model, both of which help her to orient herself without devoting much of her conscious attention to wayfinding **(see illustration on next page)**.
>
> A financial trader learning a new trading application leaves a reference dialog open on one of his monitors so that he can quickly refer to its legend whenever he is unsure of a symbol's meaning.
>
> A scientist needs the visualizations in her analysis application to always be framed by graphic scales. Without them, she finds it easy to jump to incorrect conclusions while quickly switching through different views of her lab's clinical data.

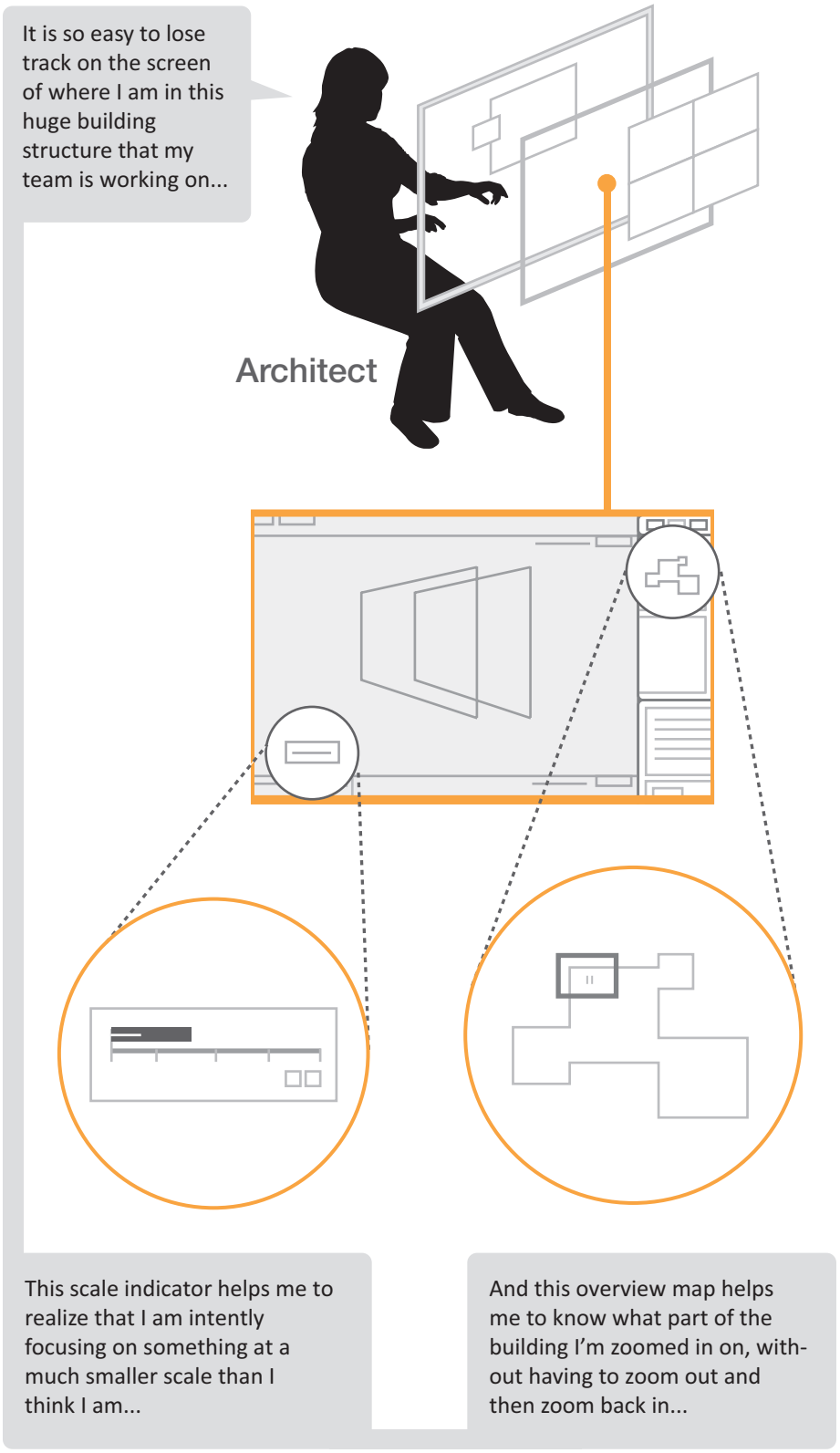
Even highly experienced knowledge workers may find that certain representational views are not self explanatory, even after extended use. Workers can benefit from, or potentially need, certain kinds of supporting content in order to make a given data display meaningful in their own practices (A).

Especially when functionality concepts contain innovative new displays (F3, F4), product teams can envision supporting information that could scaffold initial learning of abstract representations (K2). This type of content may be referred to infrequently after users have learned a tool, though it may still be highly valued when needed during ongoing interactions (C4, E1). To prevent perceptions of clutter in long term use (D4), such supporting content may be optionally viewed, with choices to present or hide it through progressive disclosure (C3) or display customization features (C8, A9).

In cases where workers could be frequently switching back and forth between displays (F9) or using a single representation to look at different data (F8), product teams can consider how important interpretive cues and explanatory information could become meaningfully integrated into representational formats (D7). In these cases, the "supporting" distinction may be somewhat artificial.

When product teams do not actively consider potential supporting content for the abstract representations in their application concepts, a variety of issues can arise. In the absence of needed information, workers may find that resulting tools are difficult to adopt (K6). They may need to repeatedly turn to assistance outside of their focus within a display (K7, J), potentially creating and enacting work arounds to ascertain the meanings of certain screens (D2, D3). People may also commit interpretative errors without realizing (C9, G3, K5), which can be effectively impossible to prevent through application logic and can effectively decrease the overall quality and quantity of work outcomes (L1).

See also: B3, F, G6, I4, I5, J7, K


It is so easy to lose track on the screen of where I am in this huge building structure that my team is working on...
Architect
This scale indicator helps me to realize that I am intently focusing on something at a much smaller scale than I think I am...
And this overview map helps me to know what part of the building I'm zoomed in on, without having to zoom out and then zoom back in...

Key *application envisioning* questions:

What explanatory content about abstract codes and data contexts could help targeted knowledge workers to more effectively learn and actively use certain representations? How might supporting cues and information be contextually presented or made interactively available in order to clarify workers' interpretive acts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What supporting content do targeted individuals currently reference while using information representations in the operations and larger tasks that your team is striving to mediate?

How might existing interpretive cues and explanatory information, or their underlying intents, be incorporated into your team's functionality concepts?

What advanced analogies to other types of information display might you draw upon when envisioning useful representational codes and contexts for your sketched application directions?

What additional supporting conventions could provide value in the varied representations that your team has envisioned? Could specific data displays valuably include further labels, keys, or scope indication?

How might support for certain representations relate to, or literally connect with, appropriate instructional content in your computing tool's help functionalities?

How could key instances of representational support be made parallel with your product's error prevention and handling conventions?

Where could persistent presentation of representational codes and context provide value in workers' practices?

When might targeted workers come to view supporting content as clutter after they have learned how to interpret a visual display? How might certain codes and context cues be interactively hidden or displayed in support of these scenarios?

What customizations might your team provide to allow targeted individuals and organizations to tailor representational aids to their own local needs?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G.

Clarifying Central Interactions

Valued computing tools can support knowledge workers' primary goals with truly compelling arcs of interaction.

The design of these central interactions can make or brake users' perceptions of an onscreen product.

During *application envisioning*, product teams can simultaneously consider potential design strategies at both the macro, framework level, and at the lower level of important individual scenarios.

By taking time to explore divergent directions for a product's central experiences, teams can discover important new design factors, while at the same time addressing common needs in the design of onscreen pathways.

When using a new computing tool, people do not typically weigh all of its available options equally. Instead, they may heavily weigh their user experiences within a small subset of supported work practices, giving those areas a disproportionate emphasis in their larger judgements of a tool's overall value. With this effect in mind, product teams' larger ideas about design strategy and scoping often need to be thought through at the level of these crucial interactions before their application concepts can truly be considered viable.

A challenge for teams envisioning specific interactions is to not take sketching certain details so far as to limit the breadth of their explorations. Conversations about specific functionality concepts can easily return to the relatively uncritical *straight to the details progression*, limiting meaningful concepting around the different "shapes" that key experiences might take. During *application envisioning*, sketching relatively granular interactions can mean working through design possibilities with only as much detail as is necessary to establish their key attributes and assess their viability.

As product teams move from high level models of work mediation down to sketching ideas for central interactions, they may identify some characteristic factors that often apply to computing tools for knowledge work. These common challenges and opportunities can be present whether an envisioned product is in a mature, understood genre or represents a novel, disruptive technology.

This category contains 7 of the 100 *application envisioning* ideas in this book:

G1. Narrative experiences

G2. Levels of selection and action scope

G3. Error prevention and handling in individual interactions

G4. Workspace awareness embedded in interactions

G5. Impromptu tangents and juxtapositions

G6. Contextual push of related information

G7. Transitioning work from private to public view

Product teams can use these ideas to explore concepts for effectively translating big picture ideas about work mediation into more concrete user experience scenarios. Some extra ideation around important functionality concepts can help teams drive high level considerations down to crucial interactions, without getting lost in every definition, design, and implementation detail. This additional concepting can also provide definers and designers with more opportunities to discover and model factors that could be pivotal across many of their products' interactive threads.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Defining interaction objects" (B), "Establishing an application framework" (C), and "Considering workers' attentions" (D) categories.

G1. Narrative Experiences

Knowledge workers can develop strong and useful expectations regarding how their work is initiated, progressed through, and concluded. To enhance users' experiences of their computing tools, product teams can reference workers' existing narratives or seek to establish new ones within their application concepts.

Examples from three knowledge work domains:

> A financial trader works through the process of booking trades over and over again in his trading application, expertly reading an offer, analyzing its context and value, making his decision, and accomplishing his chosen action. Each time he has finished this "story," he can confidently move on **(see illustration on next page)**.
>
> An architect learns to plan for certain steps when managing the potential chaos of proposed changes in her building modeling application. Proposed alterations to a building design are submitted by different consultants via the same shared tool, then collaboratively resolved, before being signed off by controlling members of the overall team.
>
> A scientist learns her analysis application's export process so well that she eventually structures some of her analysis approaches based on the stepwise, useful stepwise flow of exporting results from a study's database.

Knowledge workers learn and develop different narrative models that can drive their expectations and actions in certain situations. Workers' observed skills can be heavily based in the "cognitive scripts" that, roughly speaking, contain the abstracted and emotional stories behind their efforts. These narratives represent work practices as individuals' within a culture think of them, not as they may be directly observed performing them (A).

Product teams can envision ways to adapt and extend these narrative models as a part of peoples' interactions within their computing tools. For example, teams can leverage some of a community's existing narratives to intrinsically communicate how a given function could be used in a particular activity (B8, C1, K2). Sometimes workers' existing narratives do not entirely correspond to a product team's strategic ideas about mediating work or to certain sketched interactivities in their application concepts (C2, C4). In these situations, teams can envision new narratives, framed by those that workers already know, in order to provide a strong sense of initiation, progress, climax, and concluding feedback.

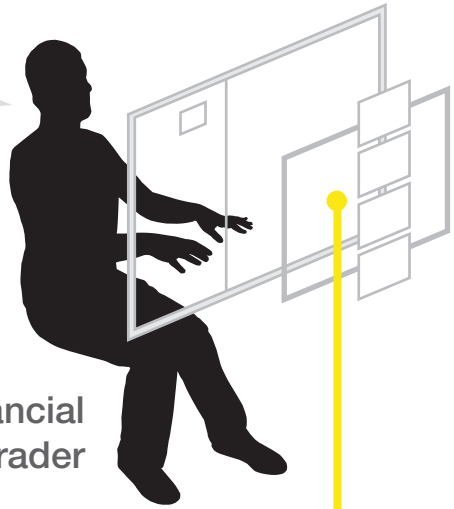
When product teams do not actively consider how narrative could play a role in their tools' potential user experiences, opportunities for applications to present a meaningful, predictable, and comforting sense of continuity can be lost. Workers may experience the inappropriately constructed or applied narratives of resulting products as limiting and mismatched simplifications (D7, K13).

Conversely, not all functionality concepts contain the same possibilities for interactive narrative. While, for example, tools intended to support standardized workflow can provide a strong narrative sense (C6), applications centered around an open, flexible workspace may not have enough preordained relation between actions to form the basis of extended and meaningful narrative structures (A5).

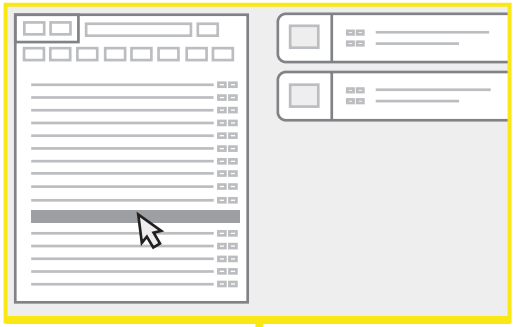
See also: C3, C7, D, F2, G, J3

Yeah, I suppose my work has a story that I repeat over and over...

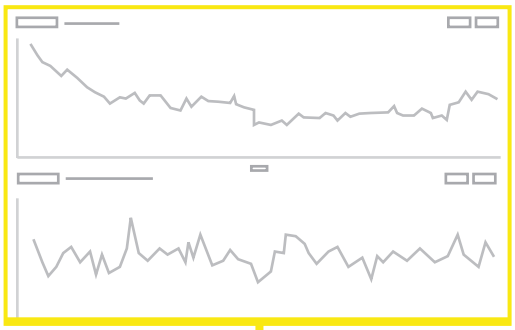
It's really several different stories, but there's one basic one that I go through again and again...



I start by choosing what needs to be tackled next. As traders, we are constantly having to rethink our priorities...



Then I analyze the potential deal that I am considering, and I think about whether to make a move and what move would be best...



And then, if I've decided to make a move, I have to quickly do the details to get it done and move on...

And that's the moment I really like. It's a very good feeling to move onward...

Key *application envisioning* questions:

How do targeted knowledge workers describe the narratives of their current work practices? How might your team's individual functionality concepts fit within these existing narratives? How might they communicate new narratives that are grounded in your sketched application's conceptual models?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How important are targeted individuals' current "cognitive scripts" in the tasks and larger activities that your team is striving to mediate?

How established and consistent are certain narratives? Do workers share very similar procedural stories in their professional cultures, or are these structures more varied within and across targeted organizations?

How have particular stories about work approaches been learned and taught within communities of practice?

How do existing narratives start, progress, climax, and conclude? Which exceptions and irregularities do they reference?

How do individuals' internal scripts encapsulate "normal," archetypal situations? How common is such normalcy in observed practice?

What useful simplifications do workers' current stories provide? How do these simplifications steer and scope effort in valuable ways?

What do existing stories leave out? What might these omissions tell your team about related automaticity, expectations of effort, tacit knowledge, perceptions of value, and other important considerations for mediating work with technology?

What pathways of interaction and progressive disclosure in your team's application concepts could meaningfully reflect and dovetail with workers' existing narratives?

What novel functionality concepts could benefit from a grounding focus in narrative structure? How might the structure of new stories help users to build appropriate conceptual models of your team's computing tool?

Which of your product's envisioned functional areas does not necessarily lend itself to preordained, "baked in" narrative structures, beyond smaller interactive expectations that are tied to certain operations?

How might your team's various narrative ideas be incorporated into your sketched directions for instructional content and other scaffolding for effective adoption?

What impact might your choices about narrative structures in your application concepts have on your product's larger design strategy and brand?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G2. Levels of Selection and Action Scope

A single interaction within a computing application can have minute or expansive consequences on stored information. To promote knowledge workers understanding the potential impacts of their action choices, product teams can envision clear levels of selection and other informative scope cues within their functionality concepts.

Examples from three knowledge work domains:

> A scientist selects a particular data point within one specific group of clinical results being displayed in her analysis application. She then applies a meaningful color code to that single point, before zooming upward to view it in the context of a much larger set of data, containing hundreds of brightly colored result groups that form massive clouds of individual data points **(see illustration on next page)**.
>
> A financial trader selects all of the components of a large trade proposal so that he can apply the same rate across each security. He then chooses a few higher value securities and adjusts their individual rates upward in order to balance out the overall deal to reflect market realities.
>
> An architect selects a specific segment of a large exterior wall in her building modeling application. She then applies a functional attribute to it, and the computing tool presents her options to either apply the same property to all wall segments tagged in the same class or to create a new class as part of applying the material trait.

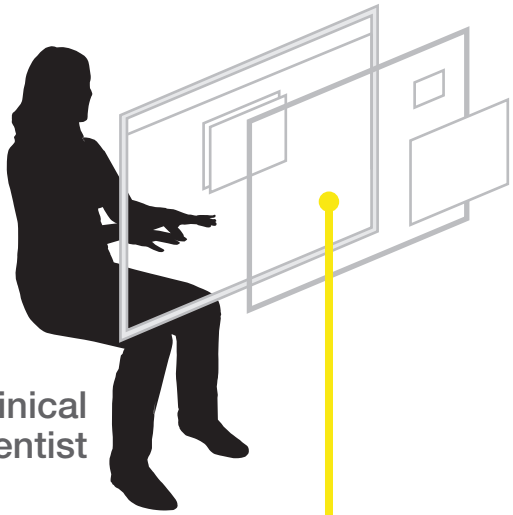
Select an object, take an action. This conventional approach to onscreen interaction can become exceedingly complex when applications are highly tailored to specialized work practices. In domain specific tools, data rich displays, relational linkages between interaction objects (B4), and other complicating factors can create situations where knowledge workers find it difficult to identify what they have selected and to predict the outcomes of certain actions.

Product teams can identify areas of their application concepts where object selection and action scope may present interaction issues (A). They can then actively envision approaches for clarifying these key cases. Depending on the character of each case, teams may define standard, learnable selection approaches based in established interaction conventions (C3, L2) or create more novel solutions to meet unique constraints (A9). Selection cues can occur at a variety of levels, ranging from an entire application view, (C2) to whole classes of interaction objects (B5, B6), to individual objects within a given representation (B1, F).

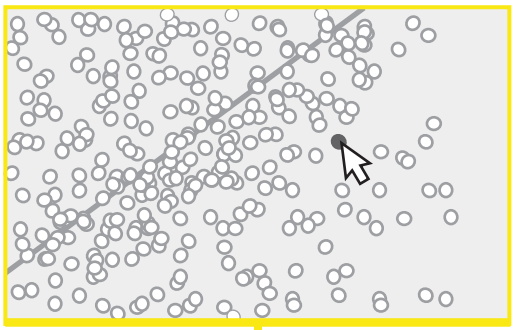
When product teams do not actively consider how clear levels of selection could allow knowledge workers to correctly choose the desired scope of their actions, resulting tools may contain seemingly straightforward interactions that lead to confusing and potentially damaging results (C9, G3) that may be difficult to recover from (H2, H3). When workers experience the act of selecting individual objects within nested or overlapping structures as effortful (D2, D3), the sense of "directness" in their interactions can become obstructed (B3, D4). These obstructions can force people to focus on obeying a computing tool's inherent rules rather than simply acting to meet their own goals, even after extended use.

See also: B, C, G, K2, K5, K6, K7

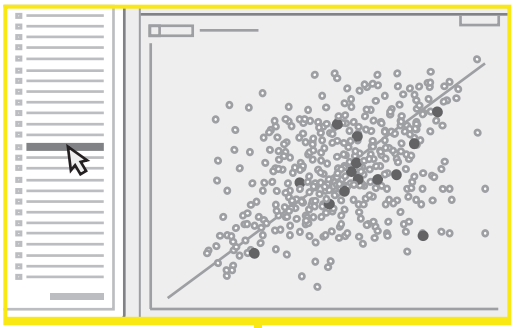
With so much data in this analysis tool, it's very important that I know what I am selecting and what I am changing...



Right now, I'm going to change the color of this single point so that its position stands out in the overall view of this data...



Or I have other useful selection options. For example, I could also change the color of this whole group of experimental results, to make it different from the many other results groupings...



Or I could change the general color of all the data points being currently displayed, which covers several different levels of data hierarchy...

Key *application envisioning* questions:

How might the complex interrelations of interaction objects in your team's application concepts be clarified into different levels of selectability? How might the potential impacts of available interaction choices be clearly communicated in different selection cases?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What new opportunities for large scale action could provide value in the work practices that your team is striving to mediate?

Looking across your sketched functionality concepts, how might you categorize the different levels of scope that a single interactive action could have?

What useful conceptual models could shape your design ideas about various levels of interaction scope?

How might identified levels of scope be clearly represented in corresponding levels of selection?

What commonly understood selection conventions could your team usefully apply throughout your application concepts?

What novel selection approaches might your team envision based on your ideas about particular, nonstandard selection scenarios?

How might different levels of selection drive the contextual presentation of targeted, goal oriented avenues of action?

What visual cues, instructive messaging, and behavioral constraints could prevent unexpected and unwanted effects that would otherwise cascade via unrecognized linkages between interrelated interaction objects?

How might an interaction's concluding feedback convey the scope of objects that it impacted? How could this messaging, in conjunction with undo functionality, turn into another form of error prevention?

How might your team envision the graphical depictions of selection cues as an overall system that is a complementary element of a larger aesthetic direction and brand?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G3. Error Prevention and Handling in Individual Interactions

Computing tools can prevent certain harmful effects of human error in specific knowledge work operations and larger tasks. Product teams can attempt to adhere to their own, internally consistent conventions across their sketched functionality concepts in order to eliminate the ability to commit certain errors, confirm workers' intentions, and handle problems when they occur.

Examples from three knowledge work domains:

> An architect uses her building modeling application to change the attributes of a material used throughout a developing design. To confirm that she wants to make the extensive change, the application presents her with a total count of building elements that would be modified. After she has applied the change across the model, a number of areas have error symbols attached to them, indicating where building codes may now be violated **(see illustration on next page)**.
>
> A scientist lowers a threshold in her analysis application to the lowest value that the underlying analysis algorithm will allow. This interactive constraint implicitly prevents her from making an error by removing the opportunity to drop the threshold so far as to make the analysis invalid.
>
> A financial trader adds one too many zeros to a number of shares in his trading application. The tool rapidly informs him that there is insufficient quantity in the organization's holdings to complete the transaction.

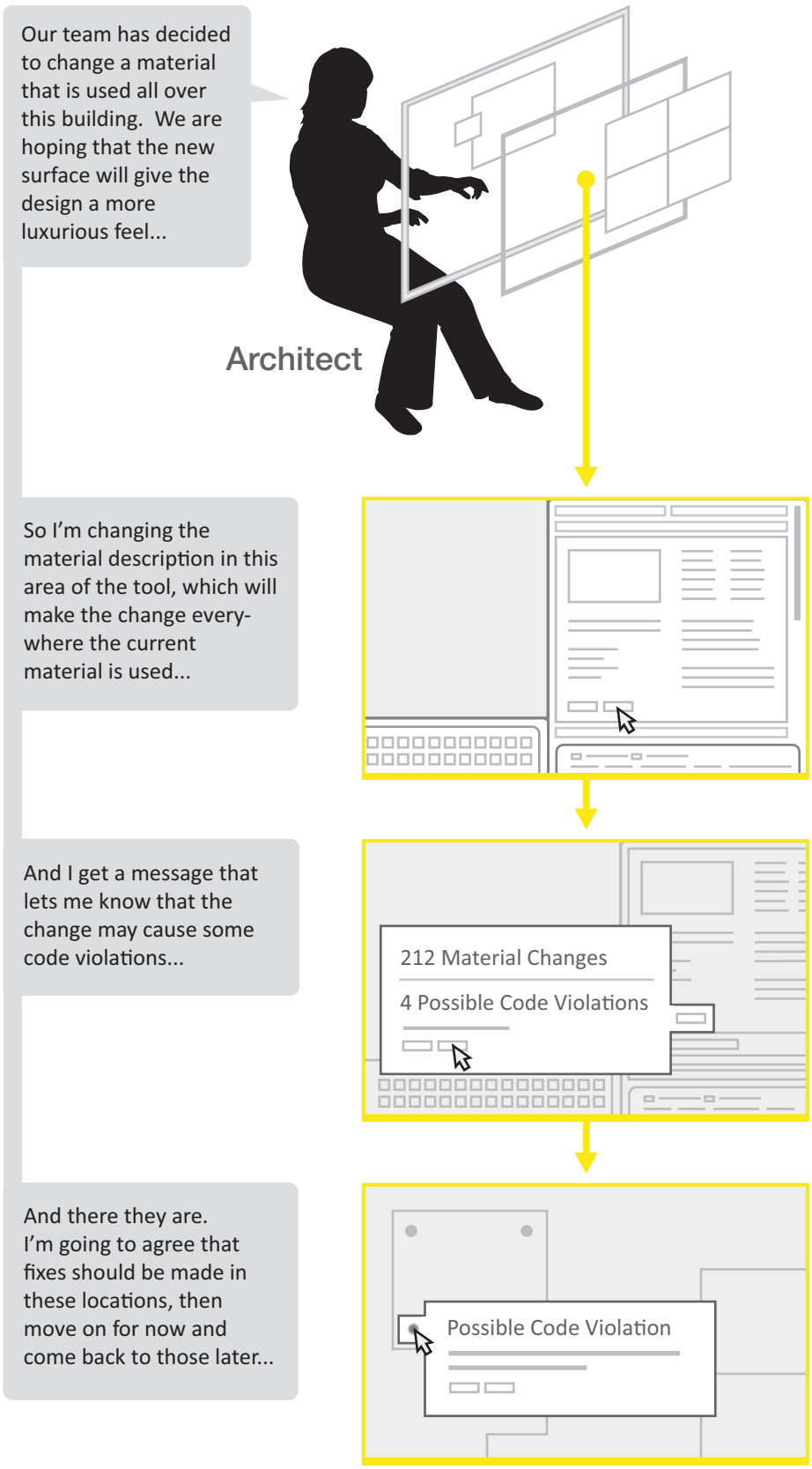
Individual actions in knowledge work can present certain dispositions for people to commit errors. Onscreen applications can reduce the incidence of some of these existing cases while at the same time introducing new error possibilities.

Product teams can investigate and identify potential cases of human error in their sketched concepts for mediating operations, tasks, or larger activities (A). Identified error cases can often be prevented or eliminated through mindful iteration of a product's behavioral constraints or by providing workers opportunities to implicitly cross check their own actions. For those error cases that cannot effectively be "designed out" of refined functionality concepts (C2, C3, K5), teams can apply their own internally consistent conventions for error prevention and handling (C9). Even when the constraints of unique error cases require novel error management solutions, teams can seek to maintain meaningful family resemblances with their tools' larger standards.

When product teams do not actively consider how damaging error cases could be usefully removed or managed within key interactions in their concepts for mediating work, resulting products may present workers with unexpected and unwanted outcomes that are difficult to recover from (H2, H3). Poor error prevention and unsatisfactory messaging can become a common complaint in onscreen tools for knowledge work (M1). Workers' perceptions of product quality and utility may also decline (K12) as they are driven to adopt defensive work arounds, such as active versioning (H1).

Conversely, too much emphasis on error prevention can be distracting (D3, D4), feel controlling, or lead to a lack of desirable flexibility (A4, A9).

See also: C10, D7, E3, G, J4, J5, K2, K6, K7, M3


Our team has decided to change a material that is used all over this building. We are hoping that the new surface will give the design a more luxurious feel...
Architect
So I'm changing the material description in this area of the tool, which will make the change every-where the current material is used...
And I get a message that lets me know that the change may cause some code violations...
212 Material Changes
4 Possible Code Violations
And there they are. I'm going to agree that fixes should be made in these locations, then move on for now and come back to those later...
Possible Code Violation

Key *application envisioning* questions:

Looking within the central functionalities that your team has envisioned, what error cases could present key problems in targeted work practices? How might your team use constraints in interactive behaviors, consistent patterns and conventions, or tailored design solutions to prevent and handle these concrete situations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What error scenarios are targeted individuals currently concerned with in the operations, tasks, and larger activities that your team is striving to mediate? Why?

How do they currently prevent and handle these errors? Could these situations present opportunities for your product?

What key error cases could arise as part of specific interactions within your team's functionality concepts? What important new cases might the abstraction of interactive computing introduce?

How might you categorize the severity of each error case that you have identified? Which could lead to loss of information, unrecognized and problematic outcomes, compromised security, or collaborative conflict?

What larger design and technology trends could influence your ideas about preventing and handling classes of errors within your computing tool's various interaction offerings?

Especially for potentially serious cases, how might your team redesign your sketched functionality concepts in order to effectively remove the possibility for errors?

What cross checks could allow workers to actively prevent errors on their own behalf?

How might the larger error prevention and handing conventions you have envisioned for your application concepts apply to specific error cases?

Could any individual error cases benefit from or require novel error management solutions that fall outside of your internally consistent, top down patterns? How might these solutions reference the aesthetics and tone of your larger standards?

Where could recovery from errors be supported solely through undo functionality, rather than more attention demanding methods?

Where might constraining or frequent interactions with error prevention and handling features frustratingly conflict with common or local needs in work practices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G4. Workspace Awareness Embedded in Interactions

To promote valuable awarenesses among colleagues acting in shared application "workspaces," product teams can envision targeted cues in their functionality concepts that could signal the performance of specific operations and larger tasks.

Examples from three knowledge work domains:

> A financial trader see his colleagues' transactions appear in the books of his trading application. When he begins entering the attributes of a new trade, such as a security name, the roster automatically filters to show him recent trades with similar characteristics. It also displays the name of any traders in his firm that are simultaneously entering the same security into their own trade forms **(see illustration on next page)**.
>
> An architect working in her building modeling application sees distant portions of the model subtly "flash" as colleagues save changes to them. These visual notifications give her a real time overview of the frequency of others' changes, as well as the potential for iconic understanding of the general shape of incoming modifications.
>
> A scientist, reviewing the progress of current experiments for a clinical study in her lab's information management application, coincidently views which samples each lab technician is currently processing.

Knowledge workers are often highly skilled at understanding how their own actions fit into the context of cooperative and collaborative activities in their organizations. Computers can have dramatic impacts on this understanding. For example, when interactive applications become a major focus in work practice, implicit visibility and communication (J1) that was once tied to performing specific actions can become hidden or entirely lost — unless computing tools have focused functionality that promotes these specific awarenesses.

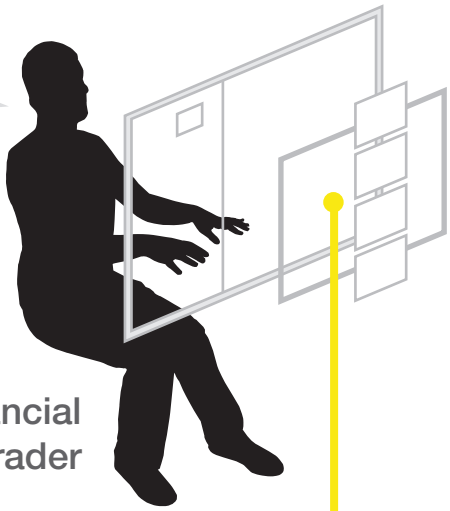
Beyond envisioning workspace awareness at the structural level of an entire application framework (C7), product teams can create concepts for specialized awareness cues in the context of individual functional areas and trajectories of action (B7, H3, J5). These targeted cues can become valuable means of offloading effort (E) that would otherwise be needed to communicate about (J) and keep track of others' interactions in a data locale.

When product teams do not actively consider how workers' actions could be made usefully and meaningfully visible to other actors within their shared application environments, resulting products may leave users with the feeling of constantly "stepping on each others toes." While an application framework may alert them to the general presence of others, without specific visibility into collaborators' actions, users may find that they have difficulty planning work (D3), knowing when to contact colleagues (J4), establishing representational common ground (F1, J2), and preventing conflicts (C9, G3).

Conversely, too much visibility into the actions of others can be distracting (D4) and can potentially lead to unwanted surveillance effects (A2, G7).

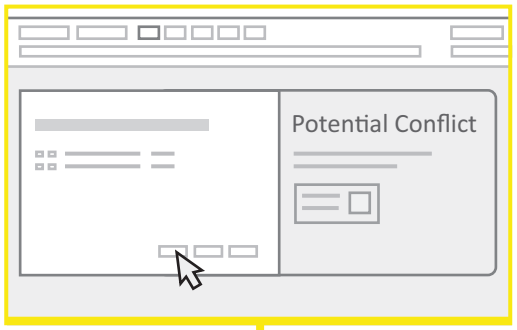
See also: A, B5, C2, C4, C5, G, K, M1

My trading tool has a lot of new features that keep me in the loop with other traders on my desk, all while I'm just focusing on my typical trading work...



If I'm on the phone and I start typing in a security name, it filters our recent deals to show me what other traders have done for that particular name and at what terms...

If I enter a security name that someone else on my desk is currently working, I get a message right there in the screen that keeps us from conflicting...



It's like that for a lot of different areas...

Like if I'm looking at how much business we've done with another firm, it will let me know if anyone else is doing that too...

Key *application envisioning* questions:

Looking within your team's individual functionality concepts, where might tailored cues about the actions of others provide meaning and value in certain cooperative work practices? What might these awarenesses feel like in practice? How might these cues reference or fit within your sketched larger approaches for workspace awareness across your computing tool's various areas?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently keep track of their colleagues' actions as part of the work practices that your team is striving to mediate?

How, specifically, do current forms of shared awareness promote the effective execution of loosely coordinated or truly collaborative work? How do they prevent conflicts?

What breakdowns currently occur due to insufficient awareness? Could these problems present opportunities for your product?

Where might the introduction of your team's computing tool remove implicit and subtle awareness cues from targeted work practices?

What larger design and technology trends could influence your ideas about how workers might remain appropriately aware of others' actions within your sketched functionalities?

How might your functionality concepts replace lost collaborative information, and potentially provide new and valuable awareness cues, as part of supporting certain knowledge work operations and larger tasks?

How might the standards set by your team's application level workspace awareness features be applied to your sketches for more granular functionalities in support of particular scenarios?

Which awareness situations could benefit from or require novel awareness cues that are substantially different than those found in the larger frameworks of your application concepts? How might they maintain a meaningful family resemblance?

Who needs to see various cues? How might awareness information relate to individuals' permissions and tailored views?

Do your sketched cues provide sufficient value to warrant their potentially distracting impacts within focused, goal oriented functionalities?

How long should specific awareness cues last? How might they be tied to longer term, stored histories for certain functions and interaction objects?

What unwanted surveillance effects could unintentionally occur from broadcasting specific actions to other workers?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G5. Impromptu Tangents and Juxtapositions

The flow of knowledge work practice can take unexpected turns, requiring sudden departures and visual referencing. Product teams can envision how their sketched application concepts could allow workers to transition between and spontaneously overlap various threads of work practice and onscreen content.

Examples from three knowledge work domains:

> A scientist is waiting for her analysis application to process a large set of clinical data so that she can visualize it. She can't remember whether she has added all of the needed data to the study file being processed, so she pauses the analysis and brings up a planning spreadsheet as a cross reference to see if all of the same sample names are present **(see illustration on next page)**.
>
> An architect is collaborating with a colleague to complete a small documentation detail in their shared building modeling application. While her colleague makes changes in real time during their discussion, the architect can occasionally shift her attention to other areas of the same building model where she needs to devote her efforts for other, separate reasons.
>
> A financial trader wants to open multiple feeds within his market information application to ensure that he is making a broadly informed decision while investigating the value of a particularly important deal.

In many knowledge work domains, the ability to make goal oriented leaps and connections can be a highly valued skill. Interactive applications can either support or impede the expression of this skill as people insightfully navigate through their work. In exploratory, synthesis oriented tasks or larger activities (A6, I5), successful knowledge work outcomes with a computer can depend on improvised, multithreaded interactions involving multiple functional areas and information displays (A8, L1).

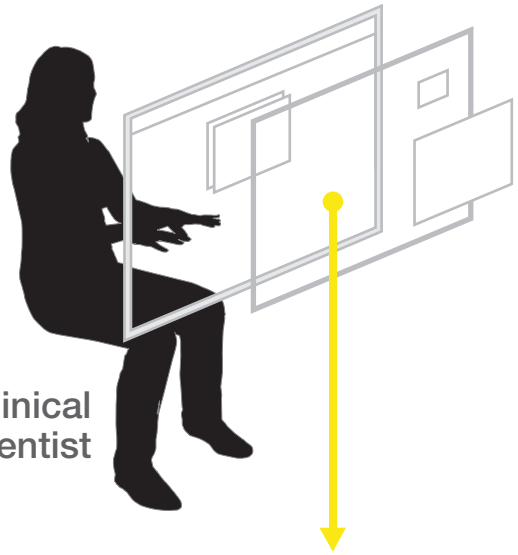
Product teams may find that the organizing pull of their rationalized models of work practice (A) can sometimes overshadow ideation that focuses on workers' less predictable tangents and juxtapositions. To help ensure that applications are not too confining, teams can identify and explore diverse scenarios for how workers' goals in targeted practices might lead to simultaneous threads of interaction and information seeking (G6, K8).

When product teams do not actively consider how knowledge workers could meaningfully stray from straightforward, idealized flows of work practice (A4), resulting applications may push users to take serial paths of action (C6) that they would prefer to conduct in parallel or in alternate sequences (A5). To accomplish their goals in these products, people may resort to effortful workarounds (D2, D3) such as simultaneously interacting with more than one instance of a tool (C2), repeatedly canceling out of processes, or frequently printing important content (J7).

Conversely, if targeted work rarely involves these kinds of improvisations (A9), then highly flexible interaction frameworks and features (C3) may detract from product simplicity, learnability (K2, K6), and interaction clarity.

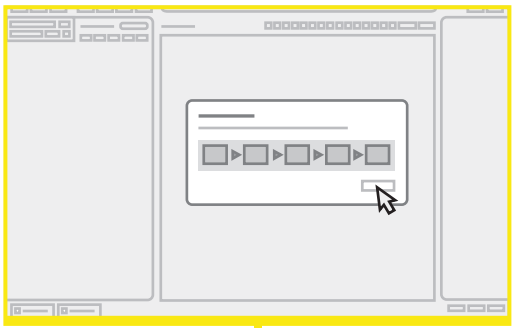
See also: D1, F5, F9, F11, G, H, I, K9, K10

So I'm just starting an analysis of a massive data set, which will take a while...



Oh, wait...

I'm not sure if I have all of the data I want in there, so I'm going to pause it...



And I'm looking at the items that are currently marked to be processed by the analysis routine...

I can't remember what I had planned, so I'm opening my electronic lab notebook and comparing its spreadsheet with what is currently listed in the analysis software...

Key *application envisioning* questions:

How might your team's application concepts allow targeted knowledge workers to freely practice the circuitous flows of their work, without unwanted structure that prevents them from valuably jumping between tasks or investigating the threads of information that they want to see? Conversely, when and where might guiding — yet limiting — interactive structure become a useful "necessity"?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What impromptu tangents and juxtapositions do targeted individuals currently make while accomplishing the work practices that your team is striving to mediate?

What value do these goal oriented excursions provide in the contexts of their tasks and larger activities?

What coordinated artifacts do targeted workers compare or act on in simultaneous, overlapping threads of effort?

What supplemental information sources do they frequently turn to during these circuitous paths? How might these sources be incorporated into your team's application concepts?

What expectations do targeted workers have about the flexibility of their computing tools? Are these expectations driven from peoples' related computing experiences, or their intrinsic understandings of their own ways of working?

What flexibilities in your functionality concepts could valuably support parallel threads of work, multiple threads of the same work, or other open variations in workers' practices?

How might the interactive flows of your team's sketched functionalities be effectively paused, stopped, and resumed?

Which of your envisioned functionality concepts do not require, or should not support, these kinds of dynamic flexibilities? What could this design stance mean for users' experiences? Should your team reconsider these limitations?

How might the interaction models of your application concepts allow targeted workers to pull up and arrange different types of information based on their moment by moment needs?

How might these flexibilities detract from the usability of your sketched product proposals? At what point could clear and functional simplicity suffer in the name of rare, impromptu edge cases?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G6. Contextual Push of Related Information

In some cases, it can be useful for knowledge work applications to adaptively incorporate "outside" feeling, potentially unexpected content into specific interactions. Product teams can envision how "pushed" domain information, presented as an optional resource, might expand workers' understanding of a subject and inform their decision making.

Examples from three knowledge work domains:

> A financial trader begins booking a deal in his trading application and is presented with a brief advisory created by analysts within his own firm. The advisory relates specifically to the item that he is about to trade, and based on its proprietary insights, he decides to cancel the deal **(see illustration on next page)**.

> An architect uses her building modeling application to label a certain area of a floor plan as kitchen space. In the tool's notifications pane, a list of defined client requirements appears with the word kitchen highlighted in each item. The pane also presents product information for appliances that need to be incorporated into the kitchen space.

> A scientist selects a specific gene variant within in a large clinical data set in her analysis application. The application then presents a small notification that provides links to recent research papers with findings related to the variant.

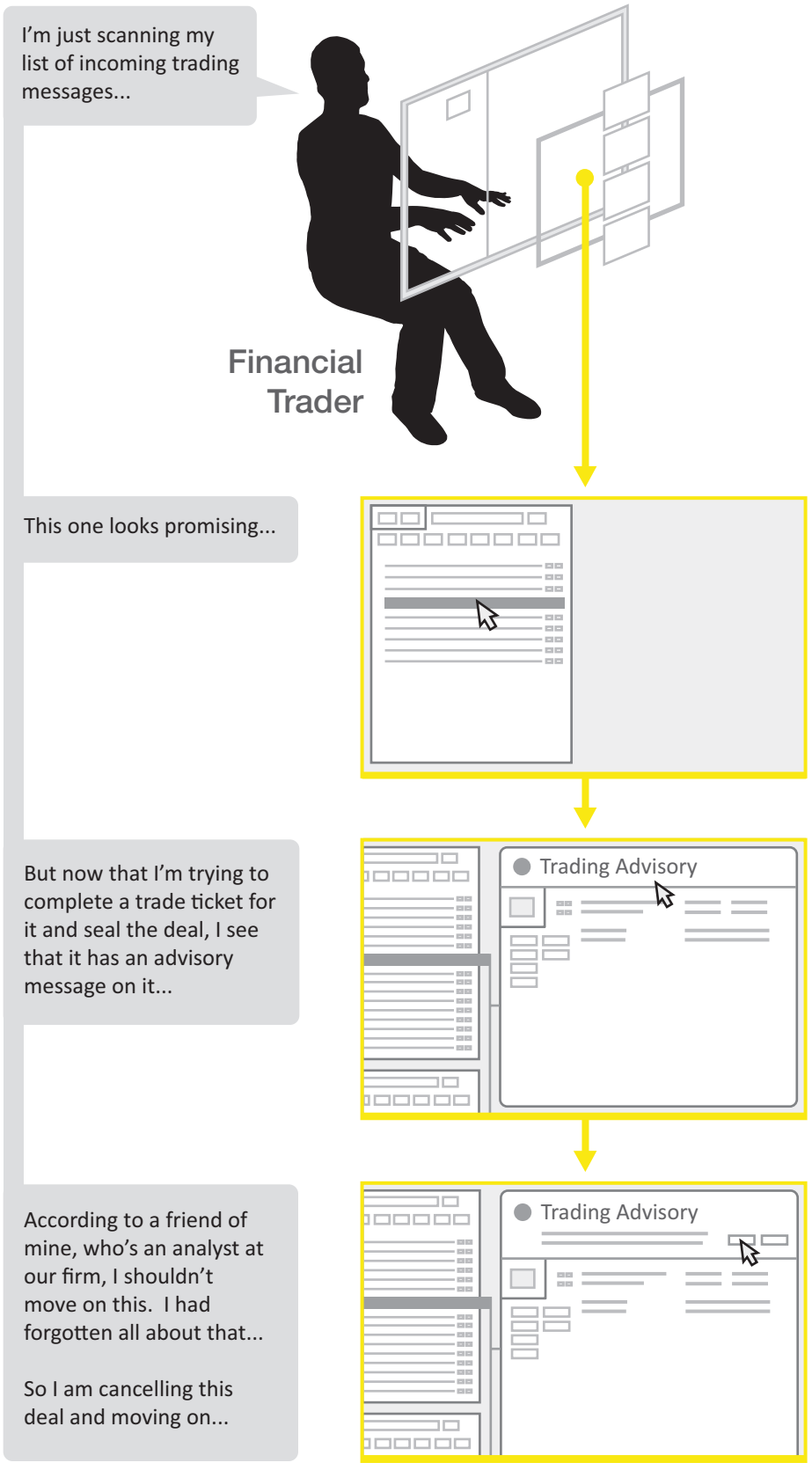
Knowledge workers in many domains struggle to keep key information in their awareness while making decisions (E1). This problem is made worse when potentially relevant information is extensive or updated frequently (I6), creating a situation where it is difficult for people to know when there could be value in seeking supporting content.

Product teams can envision targeted situations where their computing tools might provide value by "pushing" trusted information (I5, K10, K12) that is presumed to be related to workers' intents based on preceding interactions (K3). In some situations, this adaptive presentation of content can be thought of as a high level form of error prevention, aiming at complex cases that may not be preventable with strictly defined application logic (C9, G3). These information displays, rather than being just another demand on workers' attentions (D1, D3, D4), may also be envisioned as lightweight, opportunistic suggestions that may open up possibilities for individuals to make unexpected and serendipitous connections (D6, F3).

When product teams do not actively consider the potential role of supplemental information that is adaptively presented in specific interaction contexts, innovative opportunities to meaningfully support workers' synthesis and decision making practices can be lost. Highly pertinent information, stored outside of workers' typical paths, may never be seen in an influential, "just in time" way, leading to less inventive or lower quality work outcomes (K5, L1).

Conversely, if "pushed" information does not deliver relatively consistent value, people may find it distracting and try their best to ignore it (D7).

See also: A, C4, E2, E3, F11, G, I


I'm just scanning my list of incoming trading messages...
Financial Trader
This one looks promising...
Trading Advisory
But now that I'm trying to complete a trade ticket for it and seal the deal, I see that it has an advisory message on it...
Trading Advisory
According to a friend of mine, who's an analyst at our firm, I shouldn't move on this. I had forgotten all about that...
So I am cancelling this deal and moving on...

Key *application envisioning* questions:

How might your team's functionality concepts automatically incorporate useful, supplementing content into the flow of certain interactions? How might the adaptive appearance of cotextually related information positively influence knowledge workers' choices and outcomes?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently use supplemental or reference information in the work practices that your team is striving to mediate?

What specific sources do workers trust? Why?

Are valued sources created within targeted organizations, or do they come from openly available online references and outside, networked vendors?

What value do preferred sources provide in targeted individuals' decision making processes? How are they relevant?

Which sources are considered underused and could often be influential if knowledge workers took time to explore them?

Which change frequently or are too extensive to keep mentally available?

How might your application concepts meaningfully "tap into" or connect with certain preferred sources? Could the development of reference content become a service oriented element of your team's product?

When and where within your team's functionality concepts could there be value in adaptively pushing suggested content from trusted sources into the periphery of users' displays?

In which targeted operations and larger tasks could such functionality become a persistent, unwanted distraction?

What programmatic logic could effectively search for supplemental information that "fits" various situations?

How attention grabbing and directive should pushed information be, given the nature of the work practices that it is intended to support?

How might the relevancy of certain information sources change over time?

What customization options could allow targeted individuals and organizations to receive only their preferred content?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

G7. Transitioning Work from Private to Public View

Knowledge workers may want to work privately before moving their outputs to a place where certain audiences can access them. Product teams can envision functionality concepts that could provide users with clear methods of transitioning from private modes of working into defined "public" views and back again.

Examples from three knowledge work domains:

> An architect works on a new approach to a museum's facade in her building modeling application. She chooses not to update the version that her colleagues can see until she explicitly publishes it back to the main version of the building model, allowing her freedom to independently gestate her ideas (see illustration on next page).
>
> A scientist selects a set of clinical data in her analysis application. The data represents many months of effort from her lab's team, and she finds enormous satisfaction in finally uploading it to a larger database that contains the collective output of a number of separate research labs working on the same clinical problem.
>
> A financial trader knows that most everything he does in his trading application can end up being visible in real time to his colleagues, especially if it conflicts with any actions that they are taking. Understanding this visibility, he sometimes uses an old "quick calculation" tool to privately assess the viability of his ideas before turning to his trading application to book deals.

Interactive applications can act both as a "place" where knowledge work activity is accomplished and a channel by which it is communicated (J). Appropriate boundaries between the creative and the communicative can range from entirely blurred (C7, G4) to highly distinct, depending on the specifics of the knowledge work that a product concept is intended to mediate (A7, A8).

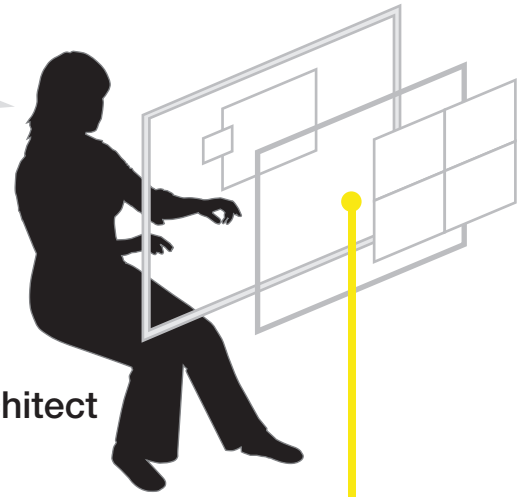
In cases where clear, separating barriers between private and public work are valued, workers may want to explicitly "move" or "send" their outputs into the view of others (C5, J3, J6, L1). Product teams can envision clear methods for managing these transitions (C4, J1), providing tailored interactions that effectively communicate a larger conceptual model of the divide between private and public "areas" or states within an application (C1, C10, K2).

When product teams do not actively consider how knowledge workers could effectively transition work from private incubation to the public sphere, resulting applications may create situations where workers do not know how to manage their own privacy (D1, D4), versions of interaction objects (H1), or effective work handoffs (J3). When people know that their interactions are visible to others in real time (E3), unwanted surveillance may limit or otherwise influence the character of their explorations of potential outcomes (H). To preserve privacy and gain the freedom to make "mistakes," workers may avoid visibility intensive products entirely while accomplishing some goals (K13).

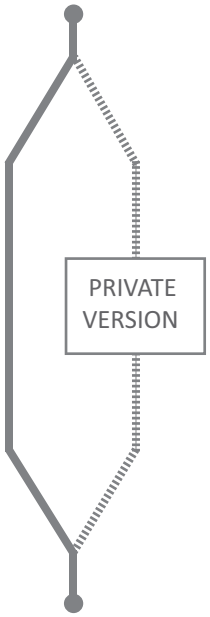
Conversely, in contexts where high levels of workspace awareness are valuable, targeted visibility into certain actions may be beneficial for work outcomes.

See also: A, B5, B6, K7, C9, G, K12, M

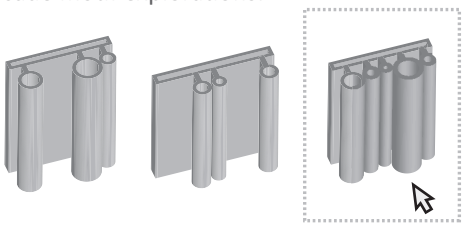
I'm still experimenting with how this facade might work, and I want some time to explore ideas before sharing with my team what I think is the best one...



So before I get started, I'm going to select an option to work on my own private version for now...



Facade motif explorations:



And now I'm merging my own version with the main building model so that the team can see the direction that I'm proposing...

Key *application envisioning* questions:

What interaction objects in your application concepts might targeted knowledge workers want to act on in private before "publishing" their efforts? What could that desirable sense of privacy mean in the context of your computing tool? How might workers recognize and change an object's current visibility — whether public or private?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently separate private incubation and public communication in the work practices that your team is striving to mediate?

Why do workers currently create and make use of these separations? What events can trigger them to bridge these boundaries in either direction?

Which "public" recipients, stakeholders, and colleagues are targeted workers typically concerned with when they think about "privacy"?

Which current work practices are often accomplished in private, either individually or within a collaborating group, and then distributed to "outside" audiences?

Which targeted tasks and larger activities are continually visible to certain collaborators?

Which efforts in workers' practices lie between the extremes of entirely private and entirely public? How do people currently conceptualize these shades of gray?

How might your functionality concepts mirror current approaches to managing the visibility of work, especially when it comes to preventing unwanted public viewing?

What reinforcing cues and messaging could clarify current privacy states within specific interactions?

How might transitional pathways between private and public states provide users with clear narratives and unambiguous feedback?

How could individual cases of the distinction between public and private collectively communicate a larger conceptual model of visibility rules within your application concepts?

How might your team's approaches for supporting the transition from private work to public visibility relate to your other concepts for supporting cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

H.

Supporting Outcome Exploration and Cognitive Tracing

Valued computing tools can play a supporting role in divergent and malleable pathways of thought and action.

Designing this kind of support requires an understanding of peoples' burdens in scenario oriented activities.

During application envisioning, product teams can map and explore areas of targeted work practices where people productively consider multiple options or "look back" through previous possibilities and choices.

By taking time to explore how users might test different scenarios or retrace their earlier cognitive paths, teams can highlight opportunities to tailor and extend their products in novel and highly useful ways.

As part of arriving at successful outcomes, knowledge workers often become highly skilled at thinking through potential approaches before pursuing a chosen trajectory. They may make explicit efforts to keep track of their various predictions and lines of thinking as they accomplish their efforts. Or they may circumstantially reflect back after some interval of time, reconstructing their pathways through an examination of their memories and the external artifacts of their actions.

Interactive applications can allow knowledge workers to externally test scenarios and react to their outcomes without committing to permanent action. Trusted computing tools can accurately store and dynamically revert to certain points within the progressions of onscreen views that flow from workers' explorations — often in greater detail than individuals can mentally visualize in their own recollections.

By removing these memory burdens and providing such externalized flexibility — outside of workers' own heads — applications can supplement users' top down thinking about their problems with rapid, free experimentation and serendipitous, chance operations. In some specialized activities, interactive simulations based on domain specific rules and information representations can transform slow and effortful practices into fluid sandboxes for thinking work.

In addition to supporting workers' exploration of outcomes, the creation and storage of interactive historical trails can be valuable for cooperative and collaborative work, recovery from errors in work practice, and evaluation of major incidents.

This category contains 4 of the 100 *application envisioning* ideas in this book:

H1. Active versioning

H2. Extensive and reconstructive undo

H3. Automated historical records and versions

H4. Working annotations

Product teams can use these ideas to explore functionality concepts for supporting, or effectively extending, workers' abilities to consider potential outcomes and retrace their interactions. Ideation focused around such support can help teams uncover innovative opportunities to effectively externalize otherwise internal work practices, potentially allowing for more creative, higher quality knowledge work outputs.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Considering workers' attentions" (D), "Providing opportunities to offload effort" (E), and "Working with volumes of information" (I) categories.

H1. Active Versioning

Actively versioning application content can free knowledge workers from concerns of damaging previous efforts while they explore alternate scenarios or otherwise advance their goals. Product teams can envision how the ability to create multiple, separate versions of interaction objects could allow workers to intentionally differentiate threads of effort and preserve milestones of progress over time.

Examples from three knowledge work domains:

> A scientist, while using her analysis application, tries to recognize key moments and jumping off points during her interactions with a set of clinical data, saving different versions of the analysis file at these important junctures **(see illustration on next page)**.
>
> An architect uses her building modeling application to create several different versions of a small building segment where she is currently thinking through detailed design. She has a few ideas of how the design could effectively play out, so she saves each idea as a named version in order to discuss them later with project stakeholders.
>
> A financial trader uses his trading application to save each round of communications on the topic of closing a large, multi-component deal. By saving each version of the negotiation, he can double check that his counterparty does not make any hidden changes to trade parameters.

Active versioning of valued information can be an important part of computer mediated knowledge work. Historically, workers have needed to version content in order to provide safety from computer glitches and to defend their progress from their own or colleagues' poorly conceived changes (H2, H3). Workers may also version content to preserve organizational memory within an activity (I7, E1, E2), to create new interaction objects from an existing object (B10), or to temporarily maintain "views" while exploring possible scenarios (G5).

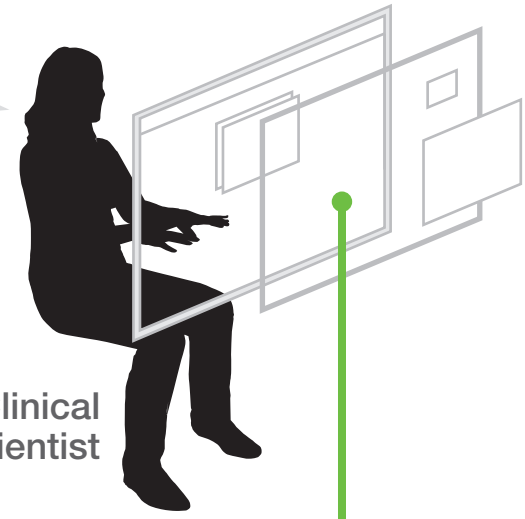
To version an existing interaction object, people often seek expected and conventional "Save as" and "Copy" commands to create named duplicates (B1, B4). Product teams can envision additional interaction pathways to tailor active versioning to targeted work practices (C4) and relevant error prevention scenarios (C9, G3).

When versioning "genealogies" are both informative in work practice and inherently complex, teams can envision functionality concepts that could allow workers to meaningfully organize (B2) and view version lineages (B4).

When product teams do not consider the potential role of active versioning in their application concepts, opportunities to support outcome exploration, cognitive tracing (H), and information security (K13) may be lost. Even though products can often leverage versioning functionality from their surrounding technology environments, knowledge workers may value integrated versioning interactivity within their tools. When they perceive available versioning methods as confusing (C1) or as excessively effortful workarounds (D2, D3), their opinions of a product's utility may suffer.

See Also: A, B2, B5, B6, E1, F1, I1, J5

At different points in my long and circuitous data explorations, I like to create separate, safely saved versions of whatever analysis that I'm working on...



- Version to save important milestone
- Version before trying new approach
- Version to save important milestone
- Version prior to adding new data

So at the end of a single session of using my analysis software, I may have chosen to create multiple versions of the overall analysis file for any of a number of different reasons...

Key *application envisioning* questions:

Could the opportunity to actively "branch" or "preserve" key versions of interaction objects provide value in the knowledge work practices that your team is striving to mediate? How might the lineages of related versions be usefully displayed, allowing targeted workers to meaningfully trace sequential arcs and branching relationships?

More specific questions for product teams to consider while envisioning applications for knowledge work:

When do targeted individuals currently create different versions of information artifacts in their work practices? What value do these versions provide?

How do knowledge workers and their organizations keep track of multiple versions of an object? What language do they use to categorize and describe them? Do they track "genealogies" across versions?

Based on your understanding of current practices, which of the interaction objects in your team's application concepts will targeted workers likely want to actively version?

How might the introduction of your computing tool create valuable opportunities for workers to use versioning to intentionally differentiate threads of effort or preserve milestones of progress over time?

What larger technology and market trends could influence your team's ideas about active versioning of stored content? What related aspects of targeted IT infrastructures might your team use as grounding for envisioning these functionalities?

What events might trigger workers to consider versioning information in your sketched application concepts?

Which of the interaction objects that your team has envisioned should not support multiple versions? Why?

How might your sketched functionality concepts contextually provide versioning options in related interactive pathways and error scenarios?

What goals and practices might drive workers to investigate versioning lineages of specific onscreen content?

What representations of lineage information might clarify valuable versioning relationships?

What functionality concepts might your team envision to allow users to navigate or "restore" previous versions?

Could automated versioning, in the form of stored history about changes in interaction objects, complement or provide more value to targeted workers than active, "manual" versioning?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

H2. Extensive and Reconstructive Undo

Undo functionality can offload effort from knowledge workers to their computing tools by storing step-by-step trails of their onscreen actions, effectively freeing them from concerns of damaging previous efforts. Product teams can envision functionality concepts that could allow workers to sequentially reconstruct earlier states in their interactive applications.

Examples from three knowledge work domains:

> A financial trader, realizing that he has made a mistake, uses undo to navigate backward in his trading application, removing some changes that he had made to the parameters of a large deal. If he had already executed the trade, undo would not have worked, and he would have had to pick up the phone to try to make corrections to the committed data **(see illustration on next page)**.
>
> A scientist has applied a change to too many clinical samples in her lab's information management application. She is relieved that a single undo action removes this change across all of the affected samples.
>
> An architect working on a residential project in her building modeling application realizes that her current design approach is creating too much obstruction between two spaces. She instinctively turns to the application's undo command to sequentially remove a series of actions, watching them "peel back" to where she had been a few minutes ago.

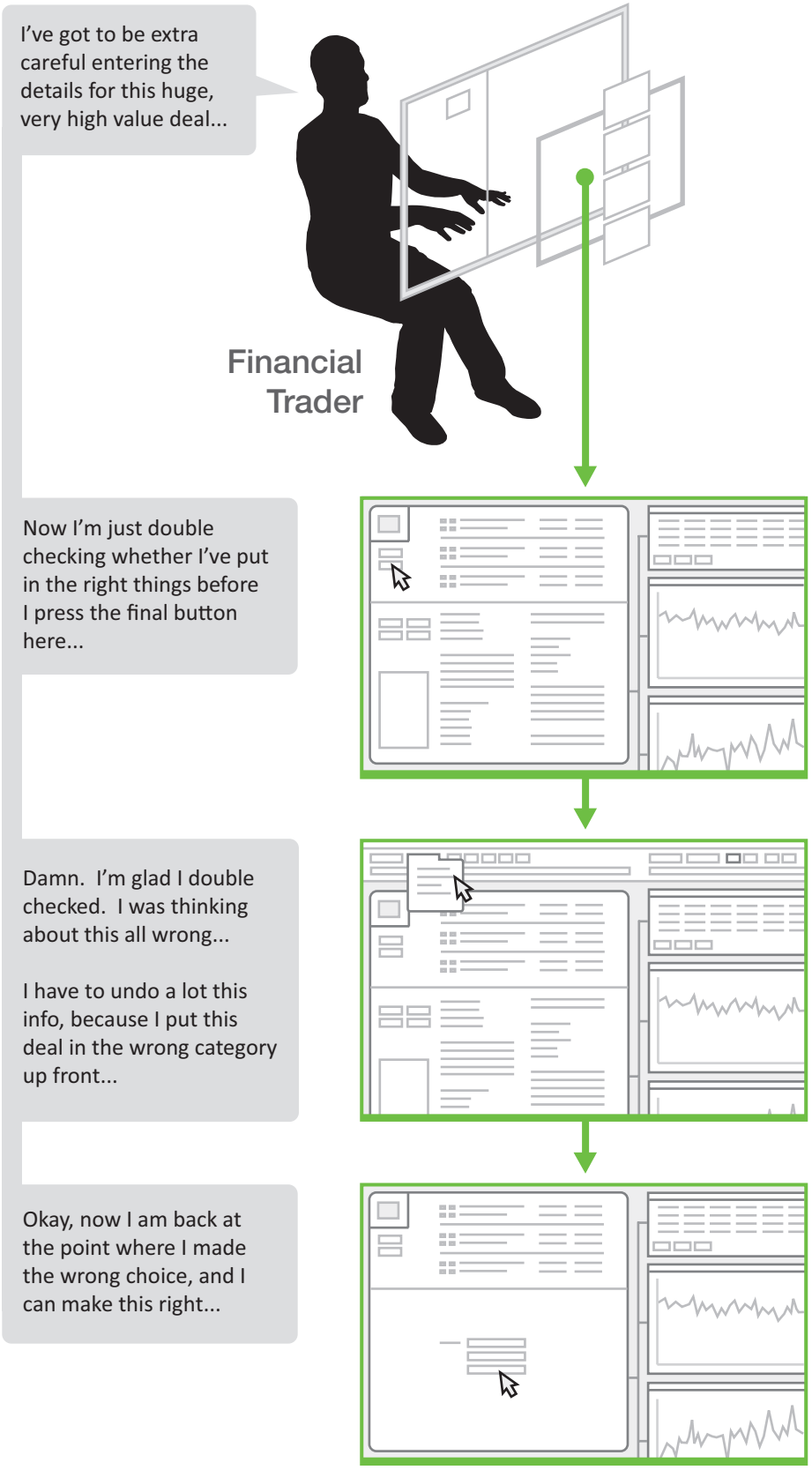
While a typewriter's correction tape leaves artifacts of its use, the digital contents of interactive applications can withstand repeated modifications without any loss of quality. Undo functionality exploits this mutability to provide valuable options.

By capturing sequential evidence of workers' previous actions, undo can become a key pathway for knowledge workers seeking to reconstruct their earlier thought processes and emergent actions (A6, E, H). Workers using trusted, stable products (K12, K13) can become comfortable with the idea of taking actions without committing to any particular outcomes that might occur. They are able to try, then try again.

In contrast, the nature of some onscreen actions should have a certain level finality. Depending on the specifics of targeted work practices (A), product teams can identify operations and larger tasks that should not be the subject of rapid, lightweight undo. When varying undo rules exist within an application concept, teams can envision clear and consistent models of how undo will operate on different functionalities (C1, H3), referencing existing conventions when applicable (C3).

When product teams do not actively consider the potential role of undo in their application concepts, opportunities to directly support dynamic and exploratory user experiences can be lost. Without undo functionalities, workers may have severe difficulties recovering from errors (C9, D2, D3, G3), potentially leading to lower quality work outcomes (L1). In an attempt to overcome these limitations, they may enact extensive workarounds, such as repeatedly saving and managing different versions of their valued data (H1, I).

See also: C2, C10, D4, D5, G5, F9


I've got to be extra careful entering the details for this huge, very high value deal...
Financial Trader
Now I'm just double checking whether I've put in the right things before I press the final button here...
Damn. I'm glad I double checked. I was thinking about this all wrong...
I have to undo a lot this info, because I put this deal in the wrong category up front...
Okay, now I am back at the point where I made the wrong choice, and I can make this right...

Key *application envisioning* questions:

How might undo functionality play a role in the knowledge work practices that your team is striving to mediate? Does the nature of targeted work allow for such uncommitted action? How might undo options "save" targeted workers from erroneous outcomes and allow them to valuably explore a breadth of scenarios?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently consider different options within different "episodes" of their work?

How can these current explorations prevent mistakes and improve work outcomes?

What expectations do targeted workers and their organizations have about undo functionality in their computing tools?

What benefits might undo provide in the specific tasks or larger activities that your application concepts are intended to mediate?

What larger design and technology trends could influence your team's ideas about what undo options could look like?

What existing undo rules and conventions might you usefully apply?

How might your product's preordained technologies influence your team's envisioning of undo experiences?

How, specifically, might undo apply within your sketched functionality concepts? What unusual cases may present definition and design challenges?

Which stored user actions might be skipped in undo sequences?

How might scenarios around cooperative or collaborative efforts impact your ideas about undo rules in shared workspaces? Should users' actions become more or less permanent in these scenarios?

What might the experience of "rolling back" actions be like after elapsed intervals of time?

What novel interactive and representational approaches might your team envision to allow workers to more effectively use their stored action histories?

How might the stored pathways of undo functionality relate to any other longer term, historical information about interaction objects in your application concepts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

H3. Automated Historical Records and Versions

Knowledge work applications can automatically store information about the actions that have been performed on specific interaction objects or enacted within a given functional area. Product teams can envision concepts for usefully presenting captured historical events in ways that could allow workers to meaningfully trace, and potentially restore, system elements to earlier states and versions.

Examples from three knowledge work domains:

> An architect wants to review changes in a particularly controversial section of a building project before a client meeting. She uses her building modeling application to view the version history of the section, which allows her to browse a list of related changes that have occurred in the last week **(see illustration on next page)**.
>
> A financial trader opens a completed transaction in his trading application to see which specific actions have been taken on the deal since he closed it. Even though he cannot change anything without making some phone calls, the history info allows him to assess whether the large and crucial deal is being handled promptly.
>
> A scientist views the history of a single clinical sample in her lab's information management application. All of the actions that have been taken on the sample, from its acquisition to different steps of processing and experimental manipulation, are chronologically listed, along with automatically saved version "snapshots".

When computers are applied to knowledge work, individual users' actions can be tracked in considerable detail. In the same vein as storing interactions in support of undo functionality (H2), computing tools can usefully "informate" work activities by automatically recording, and displaying on demand, meaningful information about actions taken on particular objects, or within particular functions.

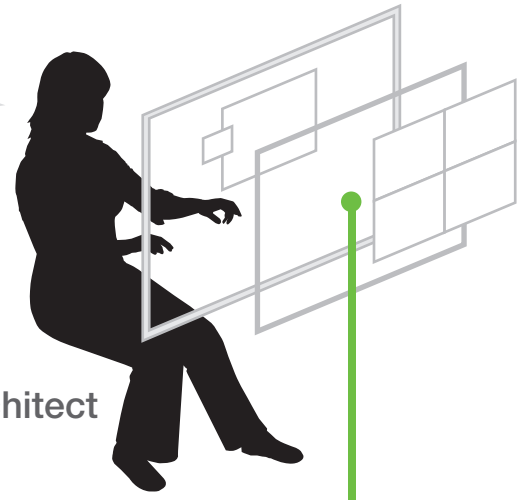
Since tracking some types of actions may not be useful or desirable, product teams can envision automated history functionalities that record only those events that may be pertinent to workers' retrospective looking goals, which may include certain actions performed by the tool itself (E5). These recorded events, or automatically stored versions, can be particularly valuable in application concepts where important interaction objects and functions are likely to be accessed by multiple workers during the course of their normal practices (A7, C7, G4).

When product teams do not actively consider the potential role of stored, accessible history for content and functional areas within their application concepts, opportunities to support valuable understandings in work practice can be lost. Additionally, when teams do not consider the possibility of automated versioning, they may be overlooking a key area of potential functional value. Certain practices may be made more difficult (D2) without automated history options, such as understanding collaborative action, communicating about work progress (J4), planning next steps (B5, D3), and evaluating critical incidents (C9, G3).

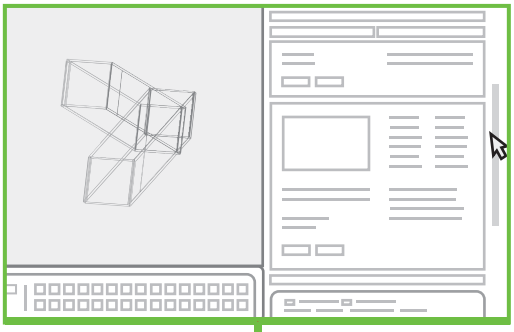
Conversely, inappropriate visibility into stored events can be distracting (D4) and can potentially lead to unwanted surveillance effects (A2).

See Also: A, B1, D5, E1, E2, G5, H

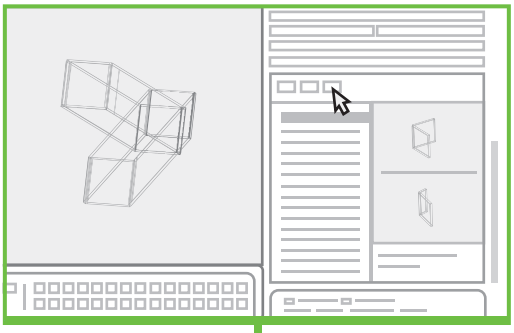
I know a lot of changes have been made in this area of the building model, and I want to remind myself about what's been in flux...



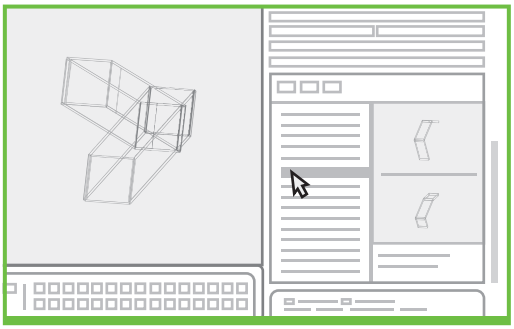
So I've selected the area in question, and I'm scrolling to show the history panel...



Where I can explore, from different perspectives, every change made within these coordinates of the building model...



And as I go through the change list for the last week, it shows useful "before and after" snapshots for each change. Alternately, I could choose to watch all the sequential changes as a video...



Key *application envisioning* questions:

When might the individuals and organizations that your team is targeting find value in looking back at what has occurred to certain onscreen objects or within particular functionalities? Why might they want to look at these histories? What related information and options — such as the ability to restore to earlier, automatically captured versions — might support their motivations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted workers currently track historical information about artifacts in the work practices that your team is striving to meditate?

What information do they record? Why?

When do targeted individuals currently refer to these historical records?

What value does historical information provide in their own efforts? In cooperative and collaborative activities?

Given workers' current retrospective looking goals and cultural environments, which specific interaction objects and functionality concepts in your team's sketched product ideas might valuably provide historical records and views?

What larger technology trends and advanced analogies to other products could inform your team's ideation of concepts for meaningful historical trails and automated versioning?

What interactive pathways could be made available from within lists of recorded events? How might these actions relate to your team's functionality concepts for active versioning and undo?

Which of your sketched functionalities, such as any significant automations, might be subject to complex, critical incidents? How might historical trails provide value after these incidents have occurred?

What novel interactive and representational approaches might your team envision to allow knowledge workers to more effectively use stored historical information about certain objects and functions?

What unwanted surveillance effects could unintentionally occur from capturing specific actions or data elements in saved historical records?

How might your team's approaches for automatically stored histories relate to your functionality concepts supporting cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

H4. Working Annotations

Knowledge workers' shorthand, contextual annotations can support their own recollections and other cognitive processes. Product teams can envision functionality concepts that could allow workers to record these lightweight, often private annotations in the context of specific interaction objects or functional areas.

Examples from three knowledge work domains:

> A financial trader, while considering different possibilities for a large trade, adds some notes to the trade form that he can look at later, when he goes to execute the deal. He wants to make sure he gives each of several different scenarios a fair testing before committing to a set of terms, and his working notes allow him to keep track of different pros and cons in order to get to the best deal **(see illustration on next page)**.
>
> An architect, having just concluded a brief client meeting, wants to make some quick notes in her building modeling application. She switches to the floor plan view of the project's file and adds a number of vague, private, reminders on building elements that require her attention.
>
> A scientist is planning a new round of experiments for a clinical study. As she thinks through the changes that she would like to make based on the outcomes of a previous round of experiments, she adds some quick notes to a number of standard templates in her lab's information management application.

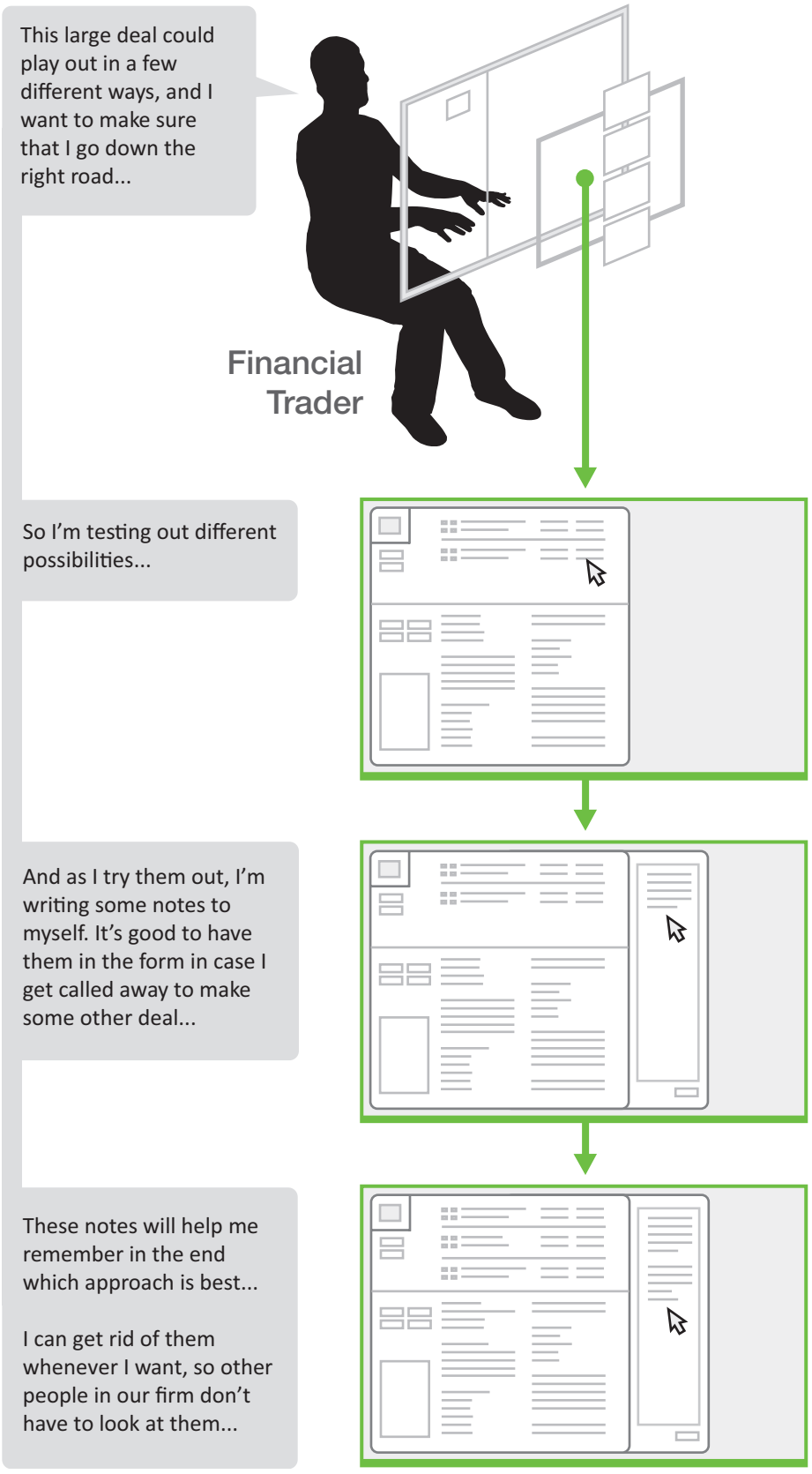
In variable, interrupted (D5), and emergent (A6) tasks or larger activities, individual knowledge workers may face some difficulty in keeping track of their own thoughts and actions. To help counter these burdens, workers may opportunistically create lightweight annotations that offload the effort of remembering their plans and situated interactions (E1, E2). These lightweight annotations can effectively tie recorded "work about the work" to specific artifacts in the annotator's practice. Although such notes may be seen within a circle of colleagues, the shorthand of working annotations may not contain fully formed ideas that are intended for outside interpretation (J1, J5).

Product teams can envision functional support for working annotations as textual notes, onscreen drawings, standardized categorical facets, attachments, links, and other options. They can also consider how these methods might come across as lightweight and informative rather than binding declarations.

When product teams do not actively consider functionality concepts for informal, personal annotations, resulting applications may drive workers to turn to outside annotation methods such as marking up printouts (J7), notepads, or other media. Individuals may find integrating these outside annotations back into their work within a computing tool to be difficult (I) and time consuming (D3, D4).

Conversely, in some knowledge work domains, working annotations may be better supported outside of interactive applications. Offline methods can provide certain affordances and a level of expressiveness that workers may find preferable to the options that are available in contemporary personal computing.

See Also: A, B2, B6, H, J1, I7


This large deal could play out in a few different ways, and I want to make sure that I go down the right road...
Financial Trader
So I'm testing out different possibilities...
And as I try them out, I'm writing some notes to myself. It's good to have them in the form in case I get called away to make some other deal...
These notes will help me remember in the end which approach is best...
I can get rid of them whenever I want, so other people in our firm don't have to look at them...

Key *application envisioning* questions:

When and where are informal, working annotations currently used in the knowledge work practices that your team is striving to meditate? How might your application concepts allow targeted workers to similarly "draw in the margins" while they work within certain onscreen displays?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which workplace locations or artifacts do targeted individuals currently apply informal, offloading annotations to as part of their work practices?

What role do these annotations play in workers' individual and collaborative mental efforts? What value do they provide?

Are working notes relatively static or are they iteratively placed and revised?

Do relatively long lasting private annotations sometimes become public communications over time? What do workers think of outside viewers seeing their working notes?

Which areas of your team's application concepts could be used in activity contexts with high levels of cognitive burden? How might users offload some of those burdens by taking advantage of direct, lightweight annotation options?

Which areas of your proposed computing tool are tied to time consuming activities where it is likely that workers will be substantially interrupted from their mental flows? How might they mark their place by using annotation functionality?

What methods of annotation could be appropriate within your team's sketched functionality concepts, based on characterized offloading goals? Might textual notes, onscreen drawings, standardized categorical facets, attachments, or links be useful?

How could your sketched representations of working annotations contextually tie them to their onscreen subjects?

Who should be able to view whose notes, based on their permissions within your computing tool?

What useful supplemental interactivity and information might your team envision around various working annotations? For example, should workers be able to check off completed notes or set durations after which their notes will fade from prominence?

How might your team's support for working annotations relate to your functionality concepts for automated historical records and versions?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I.

Working with Volumes of Information

Valued computing tools can contain massive amounts of content while somehow retaining clarity and manageability in practice.

Designing such clarity requires a critical understanding of how people think about and use certain types of information.

During *application envisioning*, product teams can map and explore their applications' potential roles in aggregating and linking to knowledge work content.

By taking time to explore potential scenarios around growing collections of stored data, teams can envision powerful, flexible, and comprehensive user experiences for information organization, discovery, retrieval, use, and sharing.

The inputs and outputs of knowledge work often amass over time within organizations. In a growing number of workplaces, networked databases and file servers now complement or have largely replaced file cabinets and physical archives as central repositories of information. Incorporating computers into work activities can increase the amount of content generated and stored during everyday work practice, since applications may track more information, at greater detail, than was previously customary.

In many knowledge work domains, how workers organize and make use of relevant volumes of stored information can have a large influence over the character and quality of their work outcomes. People often become highly skilled at cooperatively defining and using content organization schemes that are based on the tools available to them and the domain specific character of their information assets.

Interactive applications can provide clear approaches that allow workers to develop and maintain highly accessible information repositories during the normal course of their practices. Useful features and structures can allow individuals to opportunistically minimize the effort that they spend organizing their own outputs and discoveries, while at the same time making it easier to locate and make use of pertinent information.

This category contains 7 of the 100 *application envisioning* ideas in this book:

I1. Flexible information organization

I2. Comprehensive and relevant search

I3. Powerful filtering and sorting

I4. Uncertain or missing content

I5. Integration of information sources

I6. Explicit messaging for information updates

I7. Archived information

Product teams can use these ideas to explore concepts for supporting work that generates or touches large aggregations of information, whether those aggregations would be stored within or outside of their computing tools. By thinking about the expansion of information around an application over time, teams can uncover opportunities for valuable features while at the same time promoting consistent approaches across the total scope of their products.

The central notion of this category is most closely related to the “Defining interaction objects” (B), “Providing opportunities to offload effort” (E), “Enhancing information representation” (F), and “Supporting outcome exploration and cognitive tracing” (H) categories.

I1. Flexible Information Organization

Individuals and groups of knowledge workers can develop useful methods of organizing the content that informs and stems from their efforts. Product teams can envision functionality concepts that could allow workers to flexibly apply classification schemes to key interaction objects and categorize information in data repositories.

Examples from three knowledge work domains:

> A scientist is organizing clinical samples in her analysis application before she starts another round of experiments in a large research study. She creates groupings that will allow her to easily select a series of tests for the same clinical subject and visualize them as different phases in the same time series **(see illustration on next page)**.
>
> An architect defines a classification method for a project's material attributes in her building modeling application so that her team can organize material options as they are added to the system. At a high level, her team has decided to categorize materials primarily by the building areas where the will be used, and secondarily, by color.
>
> A financial trader quickly browses a market information feed, applying tags to news items that he sees as potentially relevant for his firm. He knows that when colleagues search the same feed, results that have been tagged with categories by any trader within his organization will appear at the top of the list in an attention grabbing format.

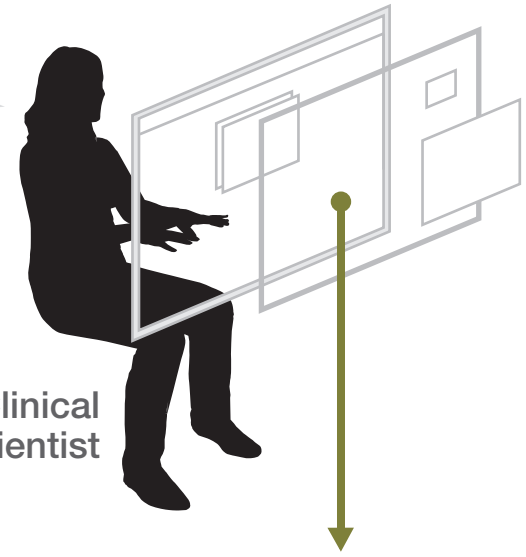
Organizing information into useful and usable schemes can be a primary skill for knowledge workers. Workers' practices may contain specific tasks or even entire activities dedicated to the act of organizing data and other artifacts. The specifics of information structures can vary widely within a domain or profession (B1, B2), with different individuals and organizations categorizing their content to reflect different conceptual models (C6) and modes of working (A8).

Product teams can envision functionality concepts that flexibly support workers' own information organization efforts (I2, I3). These functionalities can have numerous uses, such as offloading memory effort (E1, E2), supporting cognitive tracing (H), and promoting implicit communication (B5, C7, G4, J1). For known, defined schemes and categories, teams can envision a starting point of information structure, or a set of structured options, that users might then customize to meet their own local, top down needs (C8). Additionally, teams can envision bottom up organization methods that could be integral to workers' usage of certain functionality concepts, allowing users to apply meaningful categorization "along the way" to accomplishing their day to day goals (G1).

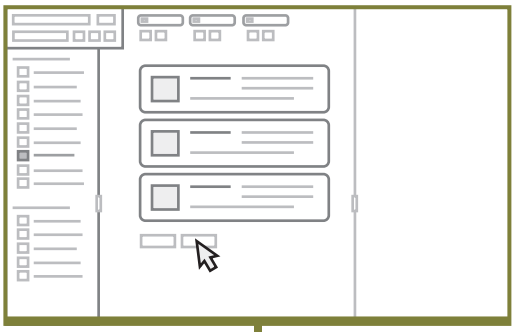
When product teams do not actively consider support for flexible information organization in their application concepts, individuals and organizations may be forced to change how they work in order to obey the implicit structures of resulting products (A4, K). This adaptation process can potentially lead to confusion and error (C9, G3), as well as to workers creating and performing excessively effortful workarounds to keep information organized in a manner that suits the established realities of their practices (D2, D3).

See also: A, B4, C5, E, F, I

The organization work, planning a large clinical study, can have as much to do with its success as all of the hours of lab work that follow...



Right now I'm creating a new set of samples in our information management tool in order to increase the volume of data collected for our lab's current project...



Next, I'm organizing the samples by dragging them into groups. These defined groupings will help later, when I'm making sense of the resulting data...

And everyone in the lab knows that each of these groupings represents a different tissue sampling time in a series of readings taken during the duration of a long clinical trial...

Key *application envisioning* questions:

How do targeted knowledge workers and organizations currently organize information in its physical form, in interactive applications, and in shared repositories? How might your team's application concepts support these existing practices while at the same time providing relevant new opportunities to classify and categorize valued content?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Who generally defines the organizational schemes and standards that people currently use in the work practices that your team is striving to mediate? Are there both top down and bottom up sources of standardization?

How do these standards vary within your targeted markets?

How have these schemes evolved into their present states over time? Historically, what forces have typically driven updates?

How frequently do classifications and categorizations change?

How do targeted knowledge workers' different roles and goal orientations currently drive different uses of the same information schemes?

How do people use the language of information categories to create common ground, facilitating collaborative sense making and action?

What expectations around information organization have workers developed from using other interactive applications?

How might the addition of your computing tool into certain work practices affect how volumes of information could be usefully organized?

What larger technology trends and advanced analogies to other domains could valuably inform your team's ideation around relevant information organization concepts?

What inherently useful organizing structures could be present in the relationships between and among your sketched interaction objects and functionality concepts?

What conventional interaction options and design patterns could allow users to flexibly create and appropriately apply needed organization schemes?

What new top down or bottom up categorization and classification options might people opportunistically make use of?

What novel functionality concepts might your team envision to decrease or remove effort that would otherwise be needed to organize information in desirable ways?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I2. Comprehensive and Relevant Search

Knowledge workers frequently need to locate stored interaction objects and onscreen information based on a variety of parameters. Product teams can envision tailored functionality concepts for specific types of goal oriented searches, as well as flexible query assembly and results representation options for unexpected and variable search needs.

Examples from three knowledge work domains:

> A financial trader uploads a list of 175 securities in order to search his firm's expansive holdings. After quickly scanning the available quantities of each item in the list, he then searches within the results to find out what proportion of current holdings for each security has been untouched for over a week **(see illustration on next page)**.
>
> An architect receives a call from a construction site about a flooring installation process. She uses her building modeling application to search for any project specification content that references "flooring" or "adhesive" in the "northern foyer." She views the results both in a table and within a 3D wireframe of the structure.
>
> A scientist users her analysis application to search a massive database shared by several cooperating clinical research labs. She is looking for only those clinical data that contain both a certain genetic marker and treatment method.

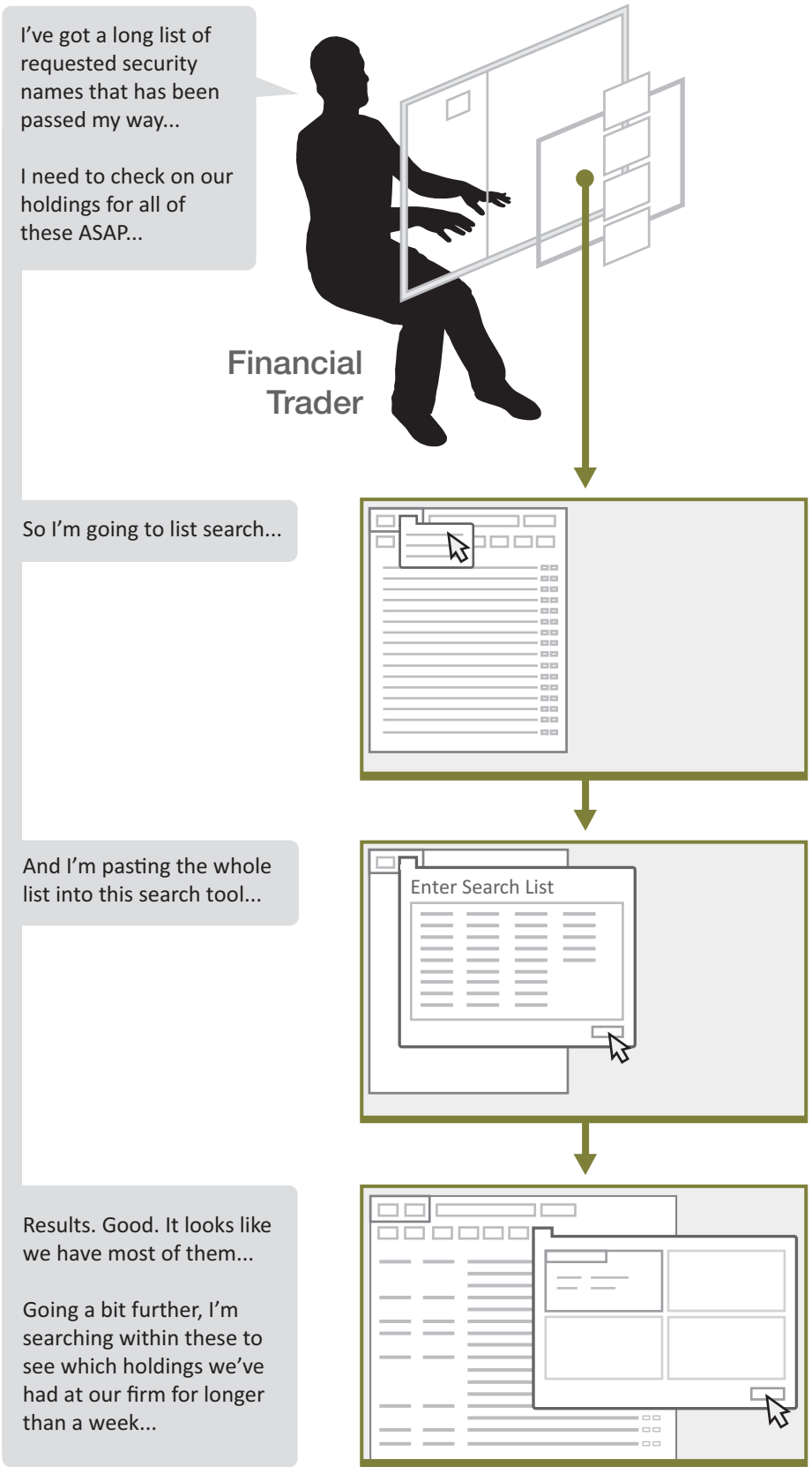
Knowing how to effectively locate specific information can be a primary skill for knowledge workers, and individuals may develop diverse searching strategies to accomplish their goals (A6, A7, A8).

The pervasiveness and utility of search in a variety of user experiences can set very high expectations. Meeting these expectations in knowledge work applications can require product teams to envision possible intersections of workers' information seeking tasks (A), diverse metadata (B2, B5, B6), integrated data sources (I5), high level algorithmic approaches, and other factors.

Beyond typical, open, textual input methods, product teams can envision supplementary search options and approaches that are grounded in the specifics of targeted work practices (B, I1). To facilitate certain frames of understanding and discovery, teams can also explore concepts for interactive results formats that complement conventional results tables (F3, F7). Thinking holistically (C4), teams can envision how using search functionality could be one operation or task within larger progressions of information seeking behavior (F4, G1), which many involve filtering, sorting, and re-representing data sets (F8, I3).

When product teams do not actively consider the potential role of comprehensive and relevant search in their application concepts, resulting tools may not adequately support workers' goals. Excluded or poorly implemented search functionality can cause people to "lose time" scanning though volumes of onscreen content (D3). Additionally, users may not locate or discover key information, leading them to incorporate less relevant items into their work outcomes (G3, K5, L1). To mitigate these issues, individuals and organizations may spend more effort in communication (J) and in organizing their data assets (D2).

See also: C8, E, F, H, I


I've got a long list of requested security names that has been passed my way...
I need to check on our holdings for all of these ASAP...
Financial Trader
So I'm going to list search...
And I'm pasting the whole list into this search tool...
Enter Search List
Results. Good. It looks like we have most of them...
Going a bit further, I'm searching within these to see which holdings we've had at our firm for longer than a week...

Key *application envisioning* questions:

Given the ubiquitous value of search functionality in many computing experiences, how might search play a useful role in your team's application concepts? What interaction objects and stored information might targeted knowledge workers be looking for as part of their work practices, and what search tools and results representations could effectively help them to find it?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What types of information retrieval and discovery goals do targeted individuals currently have within the work practices that your team is striving to mediate?

How do these goals fit within the narrative arcs of certain tasks and larger activities?

How variable are targeted workers' information seeking approaches? What commonalities might your team identify across these behaviors?

What expectations for search functionality have workers developed from using other computing tools?

What larger technology trends and advanced analogies to other domains could valuably inform your team's ideation around relevant search functionalities?

What inherent data attributes, such as the characteristics of interaction objects, could potentially be searched on in your application concepts?

Where might open, free text searching of these metadata support workers' existing information seeking goals?

How might the adoption of new computing options into targeted work practices create volumes of content that could benefit from more specific information seeking methods?

What tailored and specialized search functionality might your team envision for workers' information seeking goals? When might such "advanced" searching represent the norm, not the exception?

What novel representations of search results might your team sketch with the goal of allowing workers to meet their information retrieval or exploration goals more directly and accurately?

How could the underlying algorithms of your search concepts create content biases that could match workers' information seeking mindsets?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I3. Powerful Filtering and Sorting

When confronted with large sets of information, knowledge workers frequently benefit from the ability to reorder, highlight, or exclude specific categories of stored content. Product teams can envision functionality concepts that could allow workers to perform valuable data manipulations based on goal oriented criteria.

Examples from three knowledge work domains:

> An architect views a table in her building modeling application that lists all of the named interior features within a building design. She then filters the list to view only those items that have changes pending approval, and the 3D building representation "lights up" with colors that correspond to different approval states **(see illustration on next page)**.
>
> A financial trader filters a shared table of today's trades so that it only shows his transactions. He then sorts the table by dollar value to get a general sense for his cash volume.
>
> A scientist transforms a visualization of clinical data in her analysis application to show only data points from one group of subjects. Data belonging to subjects in the study's other four experimental conditions disappear from view, revealing an interesting visual trend.

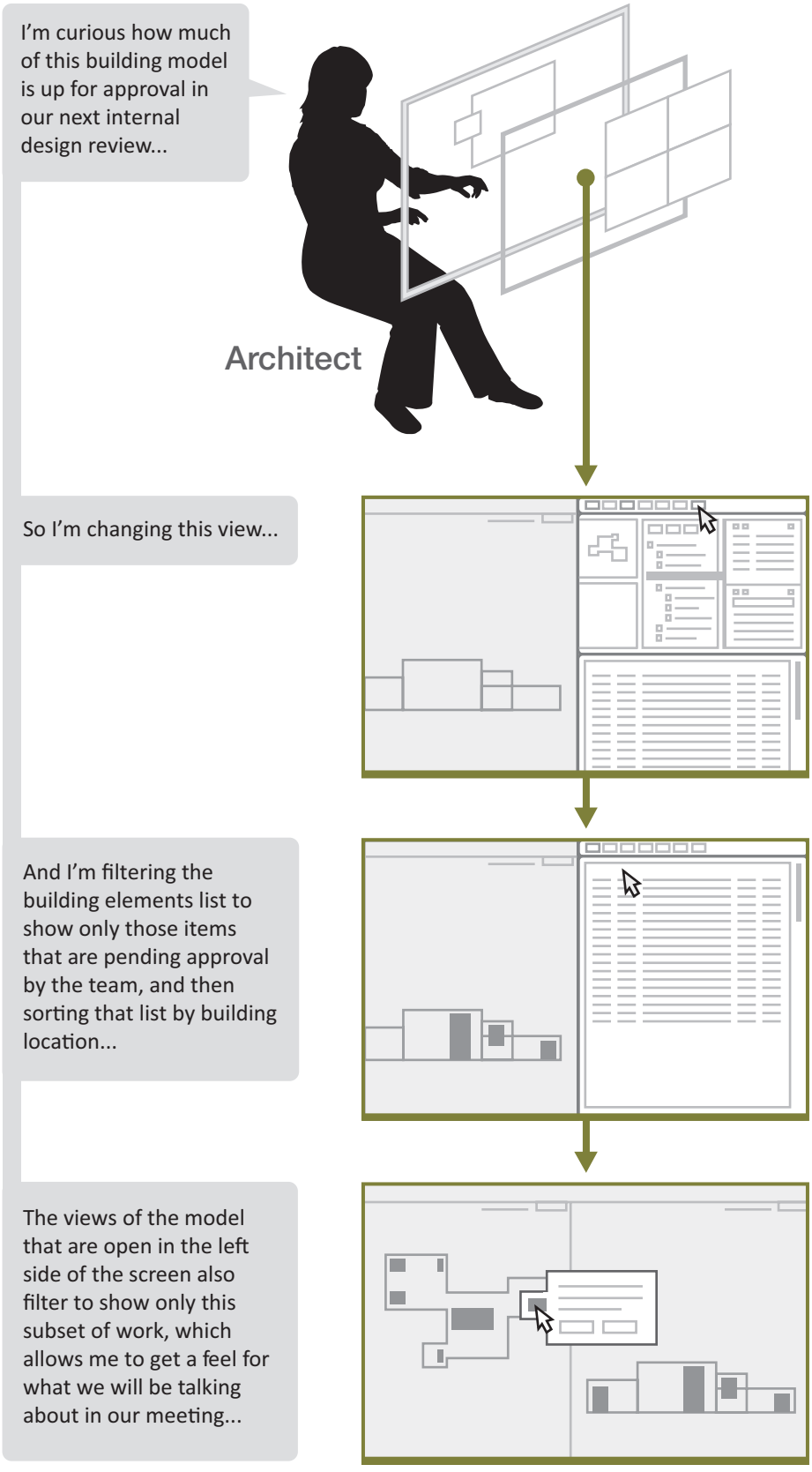
Adopting computing tools to organize and store information in knowledge work can remove useful cues and context. With so much information carrying a similar, default visual weight (C8), onscreen aggregations of content may seem somehow "flat" and overwhelming.

Product teams can envision functionality concepts for reordering large volumes of content, highlighting specific items within a content pool, or excluding information that does not match criteria relevant to workers' current goals (A6, A7, A8). Options and categories for sorting and filtering concepts can arise either from the specifics of targeted work practices (A) or from novel uses of available data (B). Thinking holistically (C4), teams can envision each of these view manipulation methods (F8, F9) as individual operations and tasks within larger progressions of information seeking behaviors (F4, G1), which may also involve search functionality (I2).

When product teams do not actively consider the potential role of sorting and filtering in their application concepts, opportunities to support workers' needs for isolating and understanding subsets of information can be lost. Individuals may find manually scanning though volumes of data to be excessively effortful (D2, D3), especially when they are familiar with the potential value of sorting and filtering options. Key categories of content may be more difficult to identify, locate, assess, and select (G2), potentially leading workers to incorporate less relevant information into their work outcomes (G3, K5, L1).

Conversely, without appropriate feedback cues (D6, F10), filtering and sorting functionality can lead to errors when users do not recognize or remember that these options are currently being applied to their views of a data set (C9, E5).

See also: E3, F, H, I


I'm curious how much of this building model is up for approval in our next internal design review...
Architect
So I'm changing this view...
And I'm filtering the building elements list to show only those items that are pending approval by the team, and then sorting that list by building location...
The views of the model that are open in the left side of the screen also filter to show only this subset of work, which allows me to get a feel for what we will be talking about in our meeting...

Key *application envisioning* questions:

Beyond, or in addition to, search options, what manipulations of application data might targeted knowledge workers value in the context of their information seeking and sense making goals? What functionality concepts might your team envision to allow workers to usefully rearrange and meaningfully sift through larger sets of content?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Outside of using search options, what approaches do targeted individuals currently employ to narrow in on information within the work practices that your team is striving to mediate?

How do knowledge workers currently reorder large amounts of content in order to meet certain goals?

How might the adoption of new computing options into targeted work practices create volumes of information where some type of filtering and sorting functionality could be useful?

What expectations for reordering, highlighting, or excluding information have targeted individuals developed from using other computing tools?

What larger technology trends and advanced analogies to other domains could valuably inform your team's ideation around relevant filtering and sorting functionalities?

What inherent data attributes, such as the characteristics of interaction objects, could become useful facets for filtering and sorting of information displays?

Which of your sketched functionality ideas could benefit from specialized, contextually tailored options for sifting through and rearranging data?

What filtering and sorting options could become standards across multiple functional areas within your application concepts?

What novel interactions might your team envision to allow users to filter and sort the content of your sketched information representations? How could these methods relate to other visualizations and view transformations that you have envisioned?

How might powerful filtering and sorting options break common ground between workers or otherwise influence collaborative practices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I4. Uncertain or Missing Content

Adopting computing tools into knowledge work practice can create new ambiguities around stored data, as well as aggravate any ambiguities that were already inherent in information collections. Product teams can envision functionality concepts that could support workers as they identify, evaluate, and act on uncertain and missing content.

Examples from three knowledge work domains:

> A scientist is reviewing a visualization of a large clinical results set in her analysis application. She notices a visual distinction that seems to be indicating missing data for one subject, prompting her to view the particular subject's data table in order to make a determination of what, if anything, has gone wrong **(see illustration on next page)**.
>
> A financial trader views a historical pricing graph of a particular security in his trading application. Several intervals of the graph are flagged to show where the pricing information is unreliable due to too much variability between the three different pricing feeds that his firm uses.
>
> An architect opens a project in her building modeling application only to find that part of the 3D model has been programmatically colored red to show conflicting changes from different members of her team. A contextual message states that the segment is critically uncertain and requires resolution before any further modifications can be made.

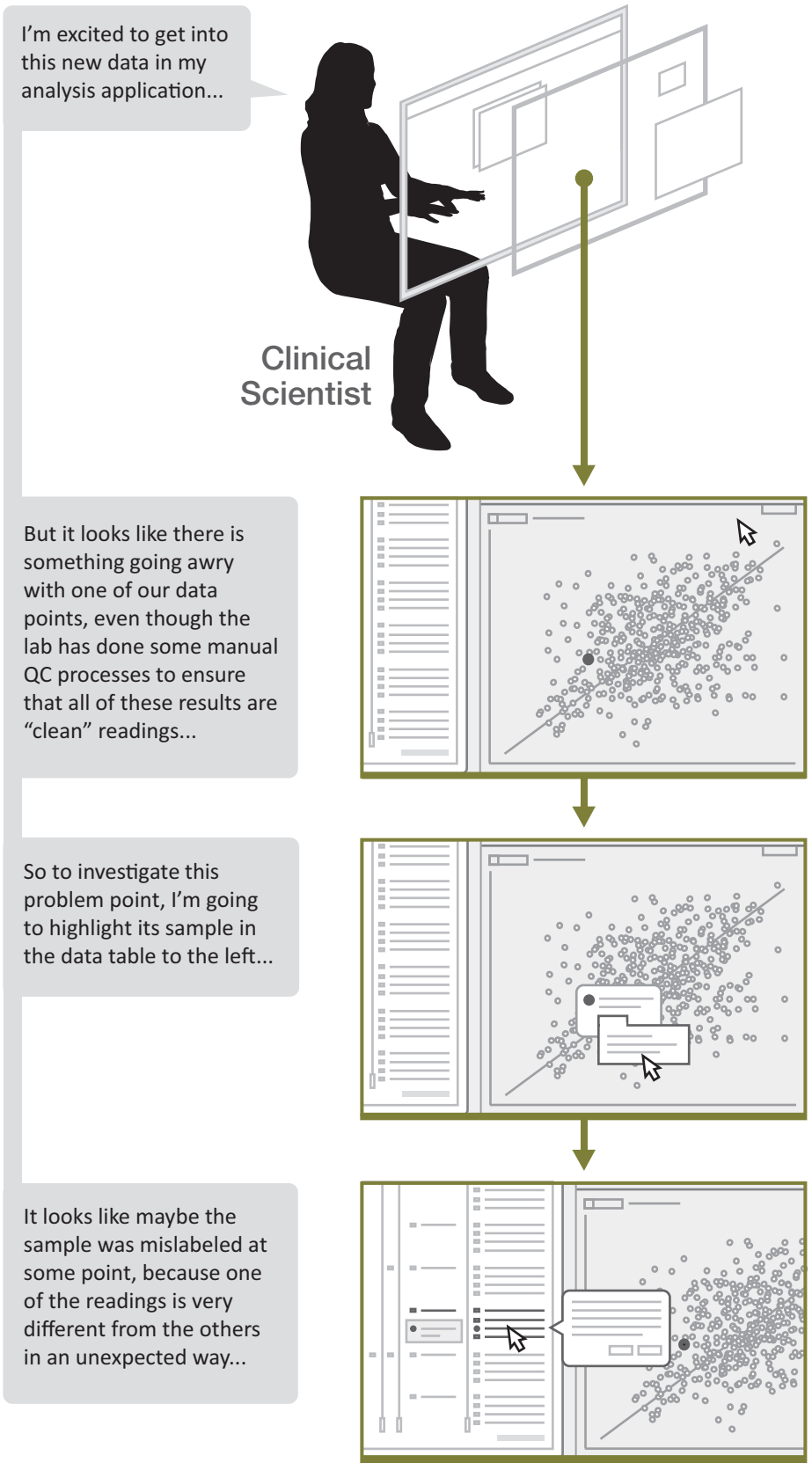
Content stored in interactive applications may present knowledge workers with a variety of surprises and problems. Large data sets can develop holes and distortions over time that workers must recognize and understand in order to act effectively. Colleagues may unexpectedly and drastically modify the characteristics of one or more interaction objects (B6, C7, G4, H3). Individual objects or entire repositories of data can be merged with like content, often with surprising results. Automations can programmatically generate garbled outputs after "choking" on small input abnormalities (E3, E4, E5).

Product teams can attempt to identify these types of scenarios in their concepts for work mediation (A). After identifying cases that are both probable and potentially damaging, teams can then envision functionality concepts that could provide workers with appropriate visibility into potential issues (F10) and clear pathways to subsequent action (C4).

When product teams do not actively consider approaches for promoting graceful handling of uncertain or missing content in their application concepts, resulting tools may promote certain types of errors (C9, G3) and lead to less beneficial work outcomes (L1). When users recognize the existence of key content problems but are not provided with tailored functional support, they may have difficulty determining how to proceed (D2, D3, K7).

Conversely, uncertain or missing content is a characteristic part of some knowledge work endeavors. In these cases, the diversity of scenarios (A6, A7, A8) may be too high for teams to envision systematic responses, and users may find any such responses to be distracting and unnecessarily limiting (A9, D4, C8).

See also: B1, B5, D6, G6, H, I, K12


I'm excited to get into this new data in my analysis application...
Clinical Scientist
But it looks like there is something going awry with one of our data points, even though the lab has done some manual QC processes to ensure that all of these results are "clean" readings...
So to investigate this problem point, I'm going to highlight its sample in the data table to the left...
It looks like maybe the sample was mislabeled at some point, because one of the readings is very different from the others in an unexpected way...

Key *application envisioning* questions:

Where might holes, conflicts, and unknowns appear in the data sets that your team's application concepts have been envisioned to import, reference, or generate? What specialized symbologies and interactive options could help targeted knowledge workers to recognize and then valuably correct — or appropriately act around — these unstable information situations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What types of anomalous information do targeted individuals currently encounter in the work practices that your team is striving to mediate?

How serious are these situations? What errors and collaborative problems can result?

How frequent are different cases of uncertain or missing information? Are they rare exceptions or a common part of what it means to accomplish the work that your team is targeting?

How do knowledge workers currently handle uncertain and missing content in their computing tools? What attitudes do they have about these anomalies in their information environments?

Where in your team's envisioned application concepts might anomalous data negatively impact work practices? How likely and damaging are these cases?

What larger design and technology trends could valuably inform your ideation around relevant solutions for uncertain and missing content?

What automated checks might your product conduct in order to determine the presence or absence of anomalous data?

What symbology and visual cues could communicate the existence of uncertain or missing content in your application's displays? What information representations and messaging could help targeted workers to effectively recognize, understand, evaluate and work through these issues?

What interaction options could be presented in conjunction with flagged data in order to allow workers to take appropriate, or even mandated, next steps?

How might your team's sketched responses for anomalous data situations relate to larger error prevention and handling conventions in your application concepts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I5. Integration of Information Sources

Some knowledge work practices involve referencing or integrating "outside" content from a variety of sources. Product teams can envision application concepts that could bring together disparate information in meaningful ways, potentially offloading effort that would otherwise be needed to navigate to multiple sources.

Examples from three knowledge work domains:

> An architect likes a feature in her building modeling application that allows her to browse diverse information based on her current selection in the computing tool. For example, it may pull up similar details from previous projects within her studio or download reviews for a particular brand of building material from a leading industry website that she frequently visits **(see illustration on next page)**.

> A scientist has selected two online databases to use as supplemental reference information about genetic sequences in her clinical studies. She links these databases to her analysis application so that when she becomes interested in a specific genetic sequence during her data explorations, she can easily compare reference information on the sequence from two different sources that she trusts.

> A financial trader knows that the market prices displayed in his trading application are an automatically blended average of three different pricing feeds, supplied by different three different vendors.

In many knowledge work domains, the quality of workers' outputs (L1) can largely be based on the information that they locate and make use of while completing their efforts. While some types of work rely on a single, standard reference, many practices can be better supported by a variety of sources, which individuals can choose to synthesize or use selectively (G5).

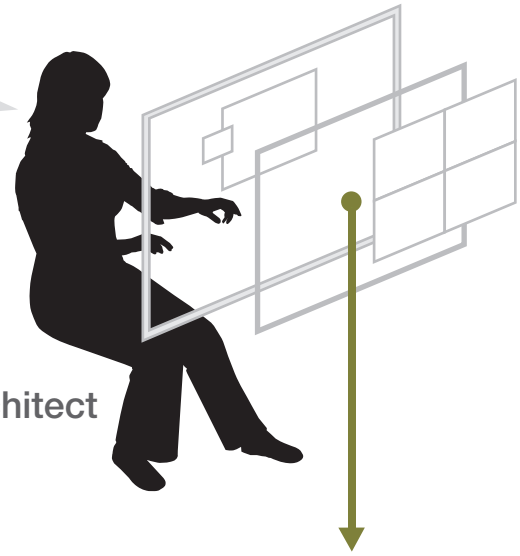
Based on specific understandings of information needs and use (A), product teams can envision functionality concepts that could positively transform how workers access content from valued outside sources (E3, E4). Integrated information can be presented in clear, potentially novel, comparative representations (F3, F5) that are tailored to workers' goals and approaches (F). When applicable, teams can also envision scenarios and design concepts for promoting the serendipitous discovery of unexpected and useful external content (G6).

Choices surrounding which information sources to use can be highly political due to conflicting industry standards and the divergent preferences of individuals or their organizations (A4, K3, K12). Appropriate levels of customization can make these integrative features more meaningful in a greater number of product adoption contexts (C8).

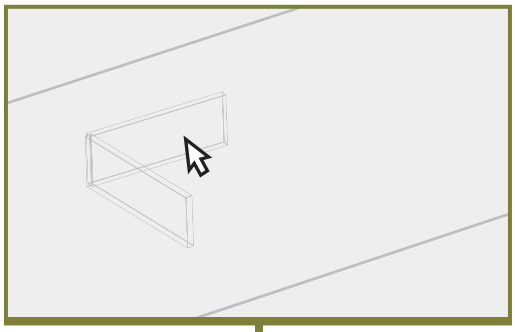
When product teams do not actively consider the potential role of outside information sources within their application concepts, opportunities to reduce workers' information seeking efforts (C4, D4) and to represent continuities across sources can be lost. When choosing to reference information feels like an extra step, people may be inclined to limit their use of outside content or to consider fewer sources, which may in turn reduce the creativity and quality of their work outcomes (L1).

See also: C1, E, I

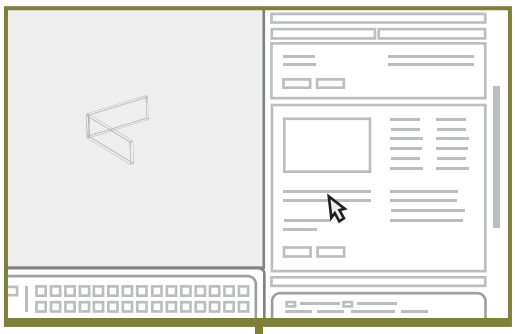
I want some inspiration around how I should change a certain part of this building model...



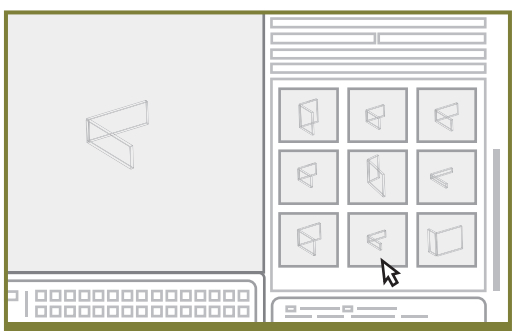
So I select it...



And this panel in the tool tells me what has currently been defined for the selection...



While this panel, which is much more inspiring, lists all sorts of similarities that the application finds. For example, it searches an open source Web database for building elements with similar forms...



Key *application envisioning* questions:

What information sources do targeted knowledge workers refer to during the specific tasks and larger activities that your team is striving to mediate? How might this content be valuably "brought inside" the bounds of your computing tool, either in its current format or in new, distilled views that are tailored to certain work goals?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which references and resources do targeted individuals know and respect? Are there standards, or are people more likely to turn to the breadth of the Web for potential options?

What value do certain information sources provide? How do workers apply their relevant content?

Do individuals currently use more than one source at the same time when seeking information? What continuities and contrasts do they look for across sources?

Which commonly used resources and data repositories might feasibly be linked to, or somehow incorporated within, your team's computing tool?

Could your firm offer its own reference content as a value added feature? What impact might such a service have on your design strategy and brand?

What larger market, technology, and design trends could influence your team's ideas about valuably making outside information available "inside" your application concepts?

When and where might certain types of integrated information provide value within the interactive flows of your sketched functionality concepts?

How might useful representational characteristics of existing information sources be preserved or even enhanced within the boundaries of your computing tool?

What novel interactive and representational concepts might your team envision to usefully distill outside information and clarify its relationships to associated content within your product? How might these displays valuably offload comparative efforts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I6. Explicit Messaging for Information Updates

Content within or associated with interactive applications can change as a result of automated updates and knowledge workers' own efforts. To prevent misconceptions and build confidence in information "freshness" and integrity, product teams can envision clear instruction and messaging around content updates.

Examples from three knowledge work domains:

> A financial trader knows that the pricing information in his trading application is updated continuously based on current market conditions. Their database of security names and symbols is less dynamic, updating every weeknight at 10 PM EST to reflect any changes announced during the previous trading day **(see illustration on next page)**.
>
> A scientist knows that her analysis application updates certain reference information about genetic sequences daily and that when she views details on a specific sequence, the information is pulled in real time from online databases, which are updated more sporadically.
>
> An architect knows that all of the reference material about building regulations in her building modeling application was entered by her team at the start of the project. This content will not be updated unless someone on her team manually makes changes, which the application will then highlight as people work on related areas of the model.

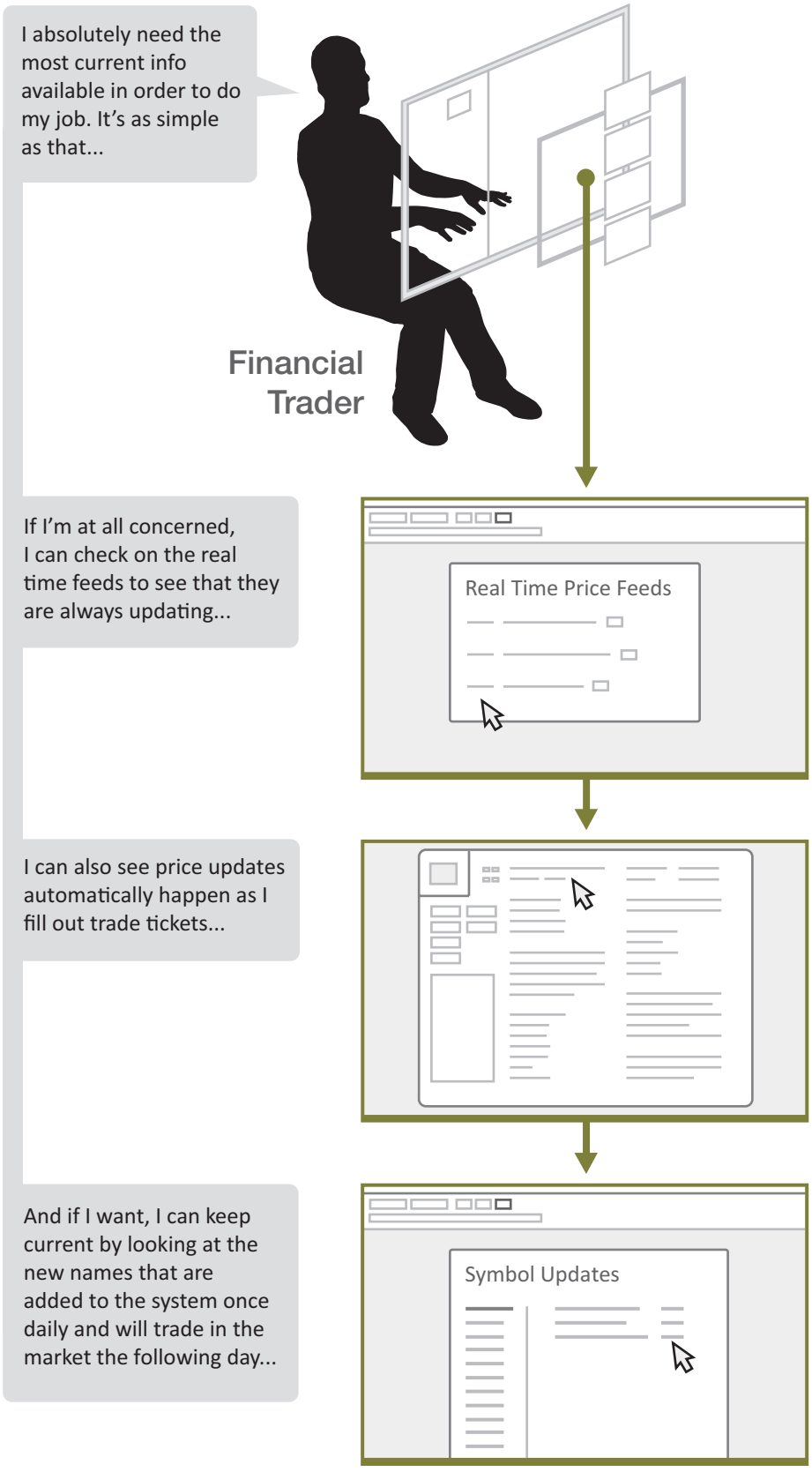
Understanding changes in information environments can be a necessary skill for knowledge workers. Individuals may monitor certain content to support their own understanding of what progress is being made inside their organizations or in their fields at large (C7, G4, H3). Their investment in maintaining an ongoing understanding of certain information's currency can also be rooted in its potential influence on their own work outcomes (L1).

The introduction of interactive computing into work practices can add new challenges for tracking these types of changes. For example, applications can increase the frequency of some types of information updates through automation (E3, E4, K10) and reduce the effort required to make sweeping changes across multiple data objects (D2, D3).

Product teams can map key information currency scenarios in targeted work practices (A) and envision functionality concepts that could support desirable awarenesses and understandings. These envisioned responses can include appropriate introductory instruction (C1, K2), notifications (D6), and contextual visual cues (F10, F11).

When product teams do not actively consider how their application concepts might promote a consistent model and messaging approach for information updates, resulting computing tools may render important transformations effectively invisible. When workers are presented with unexpected disruptions in content, a product's trustworthiness can suffer (K12, K13). Users may develop alternate conceptions of how and when content is updated, potentially causing them to act in error (C9, G3) or to incorporate extra actions into their practices to ensure that they are working with current content (D4).

See also: B2, C5, B6, C8, E5, G6, H


I absolutely need the most current info available in order to do my job. It's as simple as that...
Financial Trader
If I'm at all concerned, I can check on the real time feeds to see that they are always updating...
Real Time Price Feeds
I can also see price updates automatically happen as I fill out trade tickets...
And if I want, I can keep current by looking at the new names that are added to the system once daily and will trade in the market the following day...
Symbol Updates

Key *application envisioning* questions:

What important information used within your team's application concepts could change in ways that may be difficult to assess and understand? How might your computing tool communicate useful conceptual models and timely alerts in order to support workers' understandings of information currency?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What value do targeted individuals place on understanding the "freshness" of different types of information in the work practices that your team is striving to mediate?

What mental models and shared narratives do workers have about how the various types of information in their practices are currently updated?

How do workers assess the currency of certain types of information that they encounter in their efforts? What cues do they reference?

How can misunderstandings of information currency lead to errors? How do workers currently diagnose and recover from these errors?

How might the adoption of new computing options into targeted work practices create volumes of information where messaging about information currency could be useful or necessary?

When and where might messaging of information updates provide clarifying value in your team's sketched functionality concepts?

What guiding conceptual models might your team envision to help clarify the process behind, and potential causes of, information updates?

What initial instruction could help instill these conceptual models in users as they adopt your computing tool? How might individual interactions around information currency also convey appropriate background?

How might your team's design responses for communicating information currency relate to your error prevention and handling conventions? Your sketched approaches for supporting workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

I7. Archived Information

As activities progress over time, knowledge workers often generate information that, while valuable to their long term and organizational memories, may not need to be "present" or easily accessible. In order to improve workers' ability to focus on their current efforts, product teams can envision functionality concepts that support archiving of completed work.

Examples from three knowledge work domains:

> An architect archives all projects in her studio's building modeling application that have been fully constructed and occupied for over two years. Although there is no technical need to move old projects, everyone at her firm finds their systems easier to navigate when they contain only current and recent work **(see illustration on next page)**.
>
> A financial trader knows that he can view his group's transactions from the last three months in his trading application. Older, archived information is available on request, though he rarely needs to look back more than a week.
>
> A scientist finds that her lab is filling the capacity of their shared research database faster than expected. To make room for a massive new clinical study, she chooses to archive older studies to a separate database.

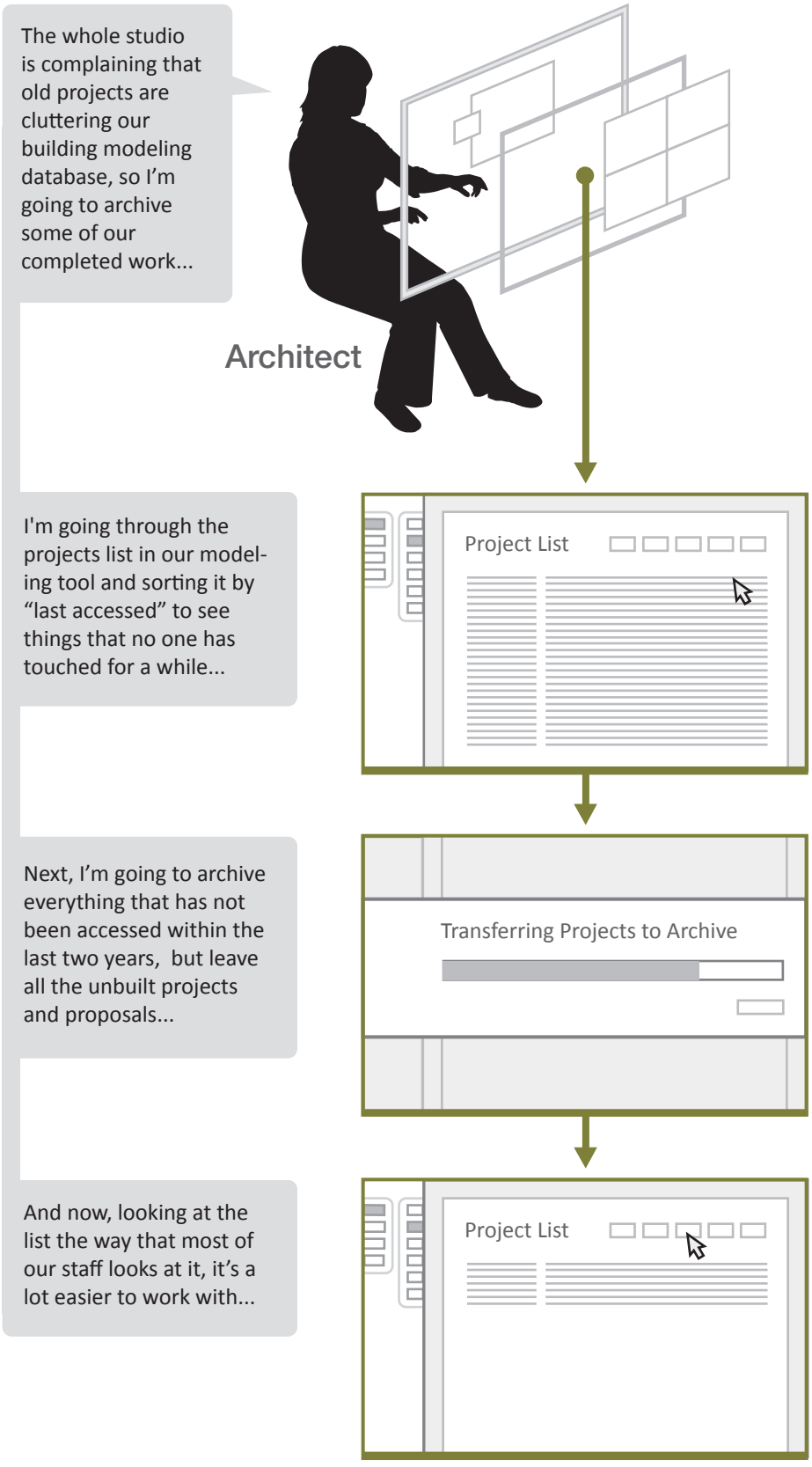
Completed projects and dated information can fill up space and cognitively clutter a knowledge work environment. Given that interactive applications can house growing volumes of information, workers may come to value the ability to exclude selected older content from their day to day onscreen views.

Product teams can envision archiving functionality based on informed predictions of how their computing tools will be used over time (A). Available options from associated storage technologies, such as off the shelf file servers or databases, may not provide compelling or effective support for specific archiving needs that can arise after product adoption (K10). Useful and usable archiving functionality can include tailored pathways for placing content into an archive (C4), managing archived content (B9, I1), searching and viewing archived content (I2, I3), and restoring archived content to an active state.

Functionality concepts for archiving can invoke the feeling of separation and distance between active and archived information (G1). Even though this feeling can be somewhat artificial in a technical sense, it may be a useful notion when envisioning instruction (K2, K6, K7) and visual representations (F10, L4) within archiving experiences.

When product teams do not actively consider the potential archiving needs that are inherent in their application concepts, resulting tools may become unwieldy through extended use. Products may present workers with escalating navigation difficulties, decreasing clarity around current workload (D3), and increasing technical performance problems (K13). Users may find the circuitous paths of their own workarounds to be excessively effortful (D2) and prone to serious errors, such as data loss (C9, G3, K5).

See also: B2, E2, H3, I, J7


The whole studio is complaining that old projects are cluttering our building modeling database, so I'm going to archive some of our completed work...
Architect
I'm going through the projects list in our modeling tool and sorting it by "last accessed" to see things that no one has touched for a while...
Project List
Next, I'm going to archive everything that has not been accessed within the last two years, but leave all the unbuilt projects and proposals...
Transferring Projects to Archive
And now, looking at the list the way that most of our staff looks at it, it's a lot easier to work with...
Project List

Key *application envisioning* questions:

What information do targeted knowledge workers implicitly or actively "leave behind" as they move forward in the work practices that your team is striving to mediate? How might your application concepts allow targeted individuals and their organizations to archive this content so that it is still available but not actively seen as part of their current efforts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How much information is currently generated in the work environments that your team is targeting? How much of that volume comes from your targeted tasks and larger activities?

Do these volumes of information eventually become an obstacle to workers' successfully accomplishing their practices?

Is the idea of archiving information already present in some way in targeted organizations? How are existing archives currently categorized and accessed?

How frequently do targeted individuals review "old" work?

What goals can trigger retrospective action? Do workers typically reopen past efforts briefly to find some specific information or do reopened items provide value over a more expansive period?

How might current archiving needs be supported within the bounds of your team's application concepts?

How might your product's preordained technologies influence your team's envisioning of archiving options? What larger technological trends could be influential?

How might the adoption of new computing options into targeted work practices create new volumes of information that could benefit from being meaningfully separated into "current" and "archived" pools?

How, specifically might "old" information "get in the way" in your team's primary functionality concepts?

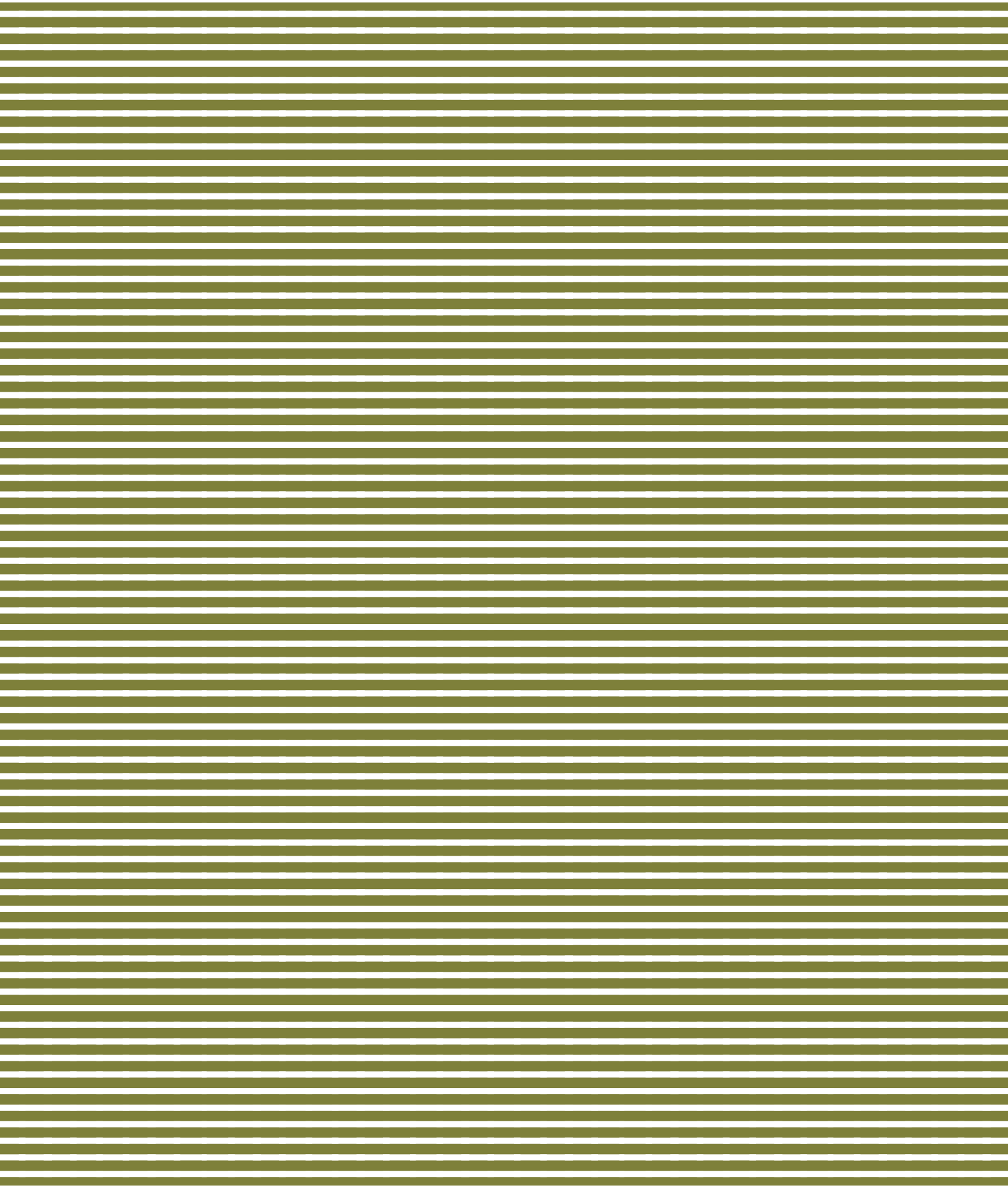
What interactive pathways might your team envision to effectively support processes of archiving information, managing archived content, searching for and viewing archived content, and restoring archived content an unarchived state?

Which of your sketched interaction objects might serve as the basis of archiving interrelated and dependent collections of stored content?

How might the states of interaction objects be used to drive archiving actions? Might "archived" be valuable as a distinct object state?

How might an archive's conceptually separate, "distant" nature inform your team's envisioning of related visual representations and instructional content?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?



J.

Facilitating Communication

Valued computing tools can enhance certain types of direct communication while opening up opportunities for more ambient and tangential signs and messages.

Designing for such meaningful interchange requires a critical understanding of where and how people deem communication to be important.

During *application envisioning*, product teams can map and explore their onscreen applications' potential roles in current and desired communication scenarios.

By taking time to think through different possibilities for interpersonal connectivity and mediated interchange, teams can uncover opportunities to tailor their functionality concepts to the conversational flows of knowledge work practice.

A knowledge worker's involvement in an activity often begins and ends with communication. In between these two points, workers' may also communicate extensively as part of the unfolding narratives of their actions.

Product teams can easily overlook potential communication scenarios that are inherent to their sketched application concepts. These scenarios can present key opportunities to more fully tie a computing tool into the larger trajectories of knowledge work activities. Without sufficient consideration of these trajectories, onscreen products can easily become isolated islands of functionality that workers must effortfully incorporate into their existing communication methods.

Given that workers may not appreciate application options that are somehow redundant with the communication channels that they are already using, product teams can envision useful methods of incorporating or connecting to targeted workers' existing channels in meaningful ways.

Beyond associations with existing channels, teams can also envision specialized functionality concepts to support well characterized communication needs, whether offline or onscreen. These functionalities might support existing practices or promote new ones, such as new forms of situated communication within an application's own information locales, either in real time or asynchronously.

This category contains 7 of the 100 *application envisioning* ideas in this book:

J1. Integral communication pathways

J2. Representational common ground

J3. Explicit work handoffs

J4. Authorship awareness, presence, and contact facilitation

J5. Public annotation

J6. Streamlined standard communications

J7. Pervasive printing

Product teams can use these ideas to explore functionality concepts that could support communication needs within the scope of knowledge work activities that they are targeting. Ideation focused on how a product could mediate existing and new forms of communication can help teams uncover innovative and genuinely valuable opportunities to more completely bridge cooperative and collaborative work practices. These bridging functions can offload certain communicative efforts and promote high comprehension experiences for workers exchanging information and thinking together.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Providing opportunities to offload effort" (E), "Enhancing information representation" (F), and "Clarifying central interactions" (G) categories.

J1. Integral Communication Pathways

Computer mediated communication can become integral to knowledge work practices, even in cases where collaborating workers and stakeholders are in close proximity. Product teams can envision functionality concepts that could provide workers with clear, relevant, direct, and contextually appropriate options for actively communicating about important application content.

Examples from three knowledge work domains:

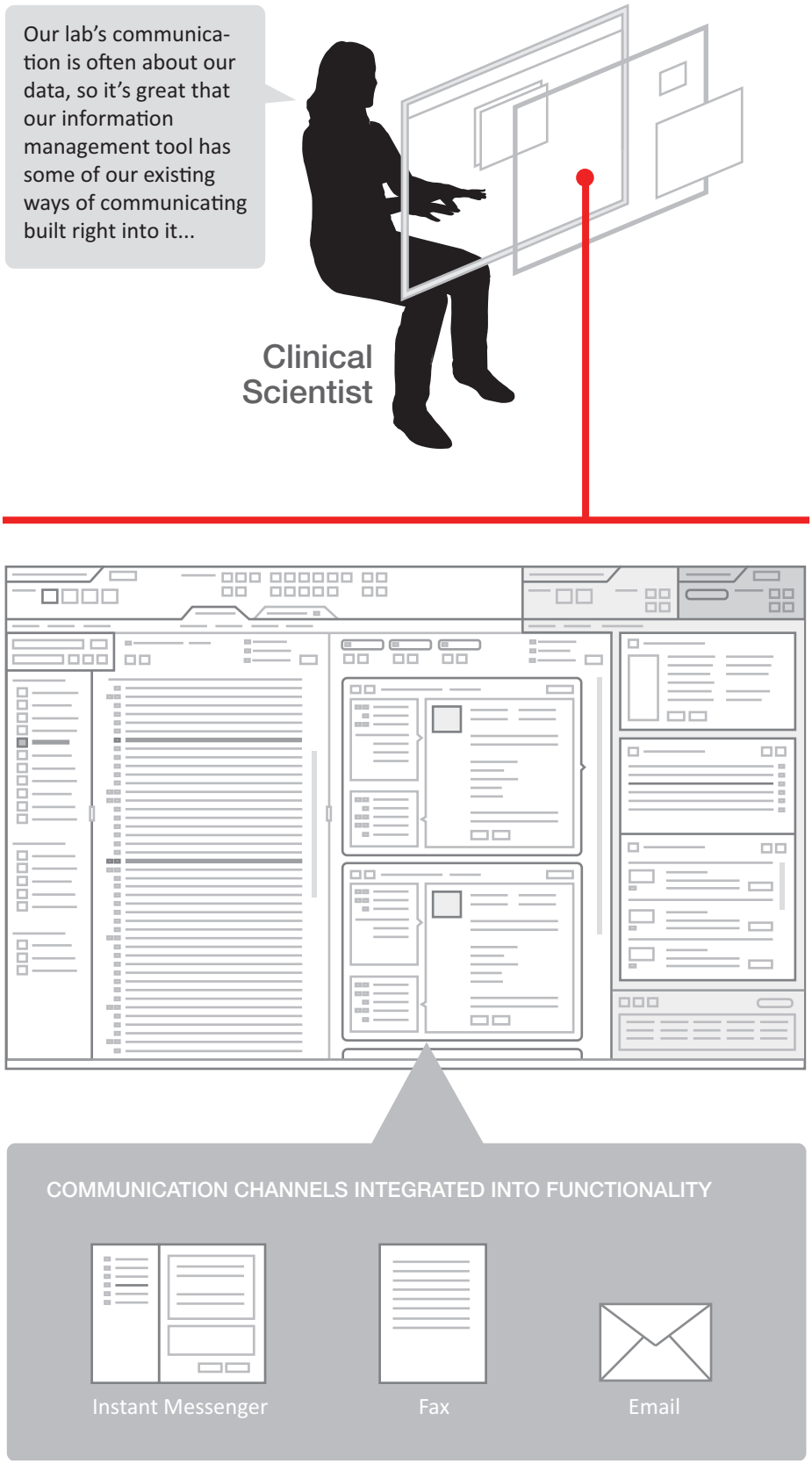
> A scientist users the integral communication options in her lab's information management application to communicate with laboratory staff, outside vendors, and distant collaborators. This integration saves some work when starting communication tasks, and it also ties communication acts and information with related laboratory data in the system **(see illustration on next page)**.
>
> A financial trader just completed a complex transaction with many parts. To ensure that the back office workers who will process the trade have the information that they need, he uses his trading application to add some special instructions to the completed trade form.
>
> An architect closes her building modeling application. As the tool shuts down, she opts to use a function that will send a status report to selected recipients on her team. This report will contain a summary of changes that she has made to the building model during her work session.

The ability to communicate effectively, in a variety of forms, is often a key part of knowledge workers' skill sets. Even though product teams may treat communication acts as separate tasks in their rationalizations of workers' practices (A), end users may not make these types of distinctions in their own mental models of their own activities. People can value applications that anticipate their communication needs and present related, contextual options (E3, E4) within the pathways of their onscreen actions.

Product teams can look for opportunities in their sketched computing tools to provide clear, relevant, direct, and integrated communication choices (C4, G1). Since not all communication tasks are the same, models of message frequency, timing, formality, importance (D3), and other dimensions can help teams envision distinct functional responses (A). These responses may involve interoperability (K8, K9) or full integration (K10) with workers' existing communication technologies. Alternately, some scenarios of communication may be better supported via new channels and options within the envisioned application itself (J5, C7, G4), rather than via existing, separate, loosely linked pathways.

When product teams do not actively consider how intentional communication could be integrated into their application concepts, resulting tools may drive users to take extra, often outside, steps in order to accomplish their goals (D2). The collective toll of these additional communication efforts on workers' productivity and satisfaction can be substantial (D3). In cases where individuals place a high value on the communicative portions of their day to day work experiences, failure to integrally support some types of communication acts may convey a lack of product quality and larger utility (K3).

See also: B, C6, D4, D6, F, J


Our lab's communication is often about our data, so it's great that our information management tool has some of our existing ways of communicating built right into it...
Clinical Scientist
COMMUNICATION CHANNELS INTEGRATED INTO FUNCTIONALITY
Instant Messenger
Fax
Email

Key *application envisioning* questions:

Why might targeted knowledge workers want to communicate about the various types of information that your team has envisioned as being part of your application concepts? With whom might they want to actively communicate? How could specific communication tasks be usefully supported through direct and integral functionality?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What trajectories of work practice that your team is striving to mediate involve intentional communication as part of their initiation or completion?

How do targeted individuals currently communicate while cooperating or collaborating around information artifacts?

What communication channels do workers use as part of targeted operations, tasks, and larger activities?

Which active communication practices do workers currently find to be problematic or tedious? Why?

What larger design and technology trends could influence your team's ideas about supporting integral communication within your product?

How might your sketched functionality concepts conveniently tie into workers' existing communication channels in goal oriented ways?

What new functionality concepts might your team envision to make communication easier and more meaningful within targeted work practices?

How might communication be different when recipients are not users of your computing tool? How could these "external" conversations remain clearly tied to your product?

How might your team's approaches for supporting integral communication relate to your other functionality concepts for supporting cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J2. Representational Common Ground

When knowledge workers collaborate around the same representations of information, their communication can require less effort and feel more direct. To support the creation of shared meaning, product teams can envision functionality concepts that could allow workers to generate and share common visual ground.

Examples from three knowledge work domains:

> An architect is having a phone meeting with an energy consultant. Since the specific design features under discussion are difficult to verbally describe, she selects an option in her building modeling application to share her view of a 3D display. The consultant, who is also using the same product, can then easily see the portions of the model that the architect is pointing at with her cursor **(see illustration on next page)**.
>
> A financial trader instant messages a particular trade's identifying information to a remote colleague. This information allows them to both have the same data pulled up in their respective applications while talking through an issue over the phone.
>
> A scientist selects a link in an email from her colleague regarding the clinical study that they are both working on. Her analysis application launches with its visualizations displayed in a way that highlights the trends that her colleague mentioned in the email message.

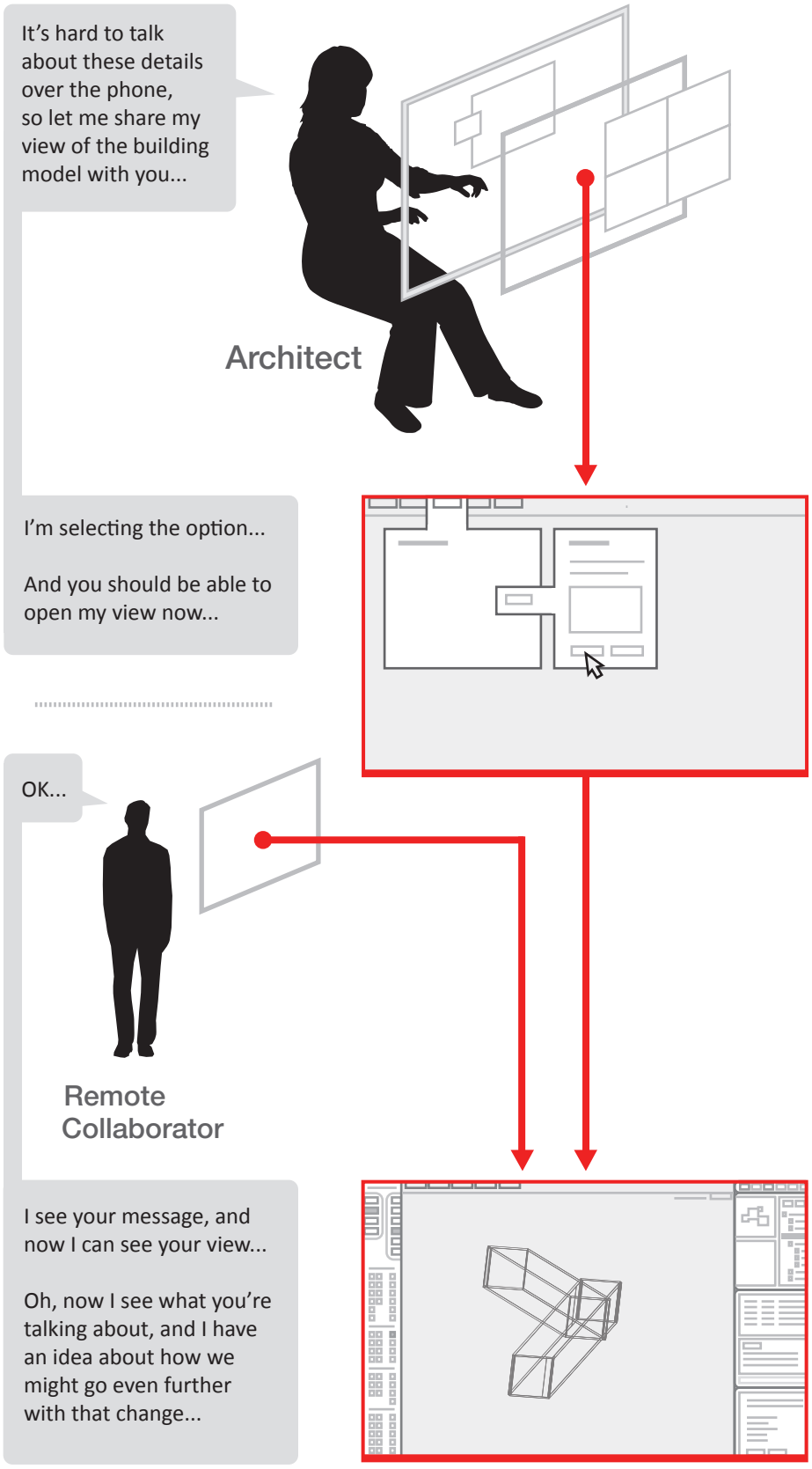
Effective communication and collaboration in knowledge work practice is often built upon a common ground of shared information representation (F1). When workers collaborate in person, they can typically establish common ground around a shared display, set of printouts (J7), notes, or sketches. Generally speaking, these representations are visual artifacts, and many of them were initially created onscreen, using one or more computing tools.

When, for any of a number of reasons, workers do not have the luxury of meeting face to face, they may attempt to establish common ground through shared communication technologies. In some cases, based on cultural norms (A1, A8), needs for message persistence (I), and a variety of other factors, onscreen applications can offer desirable methods for "gathering around" key information.

Product teams can envision functionality concepts, tailored to the specifics of targeted communication scenarios (A7), that promote useful simplicity in the act of arriving at a shared representational focus.

When product teams do not actively consider how their application concepts could support the creation of representational common ground, resulting products may decrease the ease and quality of workers' communications. While computers can be powerful tools for creating information representations (E3, E4), the highly dynamic displays and large volumes of stored data in many onscreen applications can make establishing common ground excessively difficult (D3, F8). For example, when attempting to share information at a distance, collaborators may find it effortful to retrace and verbalize the steps needed to recreate their current displays (D2, G3, H).

See also: A, B, C5, C9, E, F, G7, J


It's hard to talk about these details over the phone, so let me share my view of the building model with you...
Architect
I'm selecting the option...
And you should be able to open my view now...
OK...
Remote Collaborator
I see your message, and now I can see your view...
Oh, now I see what you're talking about, and I have an idea about how we might go even further with that change...

Key *application envisioning* questions:

What information do targeted knowledge workers currently share in order to make their exchanges clearer? How might your team's application concepts support existing approaches for creating common ground? What novel functionalities might you envision to valuably support the sharing of information views within mediated communication?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How do targeted individuals currently establish common visual ground for communication in the work practices that your team is striving to mediate?

What value do different types of common ground provide?

What information representations are commonly referenced in communication acts? What features of these representations are often important to share?

What language and gestures do knowledge workers currently use when referring to their shared information artifacts?

What breakdowns and errors in shared understanding can currently occur around these artifacts? How might the flexible displays of your team's computing tool aggravate these problems or create new ones in a similar vein?

What larger design and technology trends could influence your ideas about how content in your application concepts could be conversationally shared?

What conventional design patterns and functionalities might your team consider using in order to valuably support common ground?

What interaction objects in your sketched product directions could extend or replace the information artifacts that workers currently "gather around"?

What aspects of your functionality concepts might workers opportunistically use to create common ground? How could your team enhance these aspects to promote such use?

What novel functionality concepts might your team envision to provide workers with new options to dynamically share relevant information within their discussions?

How might your team's approaches for supporting representational common ground relate to your other concepts for supporting cooperation, collaboration, and workspace awareness?

How might common ground functionality be different when recipients are not users of your computing tool? How could these "external" conversations remain clearly tied to your product?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J3. Explicit Work Handoffs

As part of contributing to larger activities, knowledge workers often need to formally or informally handoff their efforts to certain colleagues and stakeholders. Product teams can envision communication functionalities that could allow workers to clearly and directly deliver certain tasks or interaction objects.

Examples from three knowledge work domains:

> A financial trader sends part of a list of offers to another trader in his group so they can divide up a larger pool of decision making and transactional effort, which needs to be resolved as soon as possible. His trading application notifies him when his colleague has accepted the request, allowing him to focus his efforts on the offers that are left on his plate **(see illustration on next page)**.
>
> A scientist defines a number of clinical samples in her lab's information management application. She then assigns the task of running experiments on the samples to a particular lab technician, who will receive the task description in his prioritized queue, along with links to the sample files in question.
>
> An architect finishes another version of a large, structural arch in her building modeling application, based on feedback from a civil engineer. She then uses a feature in the modeling tool to hand the component back to the same engineer for further review and modifications.

The complex problems tackled in knowledge work organizations may require the input of several or many different individuals, roles, and skill sets. Increasing specialization can mean that workers provide their inputs sequentially, with related artifacts and responsibility being passed back and forth.

Appropriate design approaches for supporting handoffs can vary based on whether they are highly structured or largely improvised. In highly structured cases (C5, C6), mediated handoffs can occur at defined points in targeted workflows (A9, C4). In improvised, emergent work scenarios (A6), options for handoffs can be provided more contextually, opening up opportunities for dynamic, shared decision making (G5, J) and the flexible offloading of effort (D2, E, A7, A8).

To promote confidence (D3, G7, K13) and awareness (C7, G4) in handoff actions, teams can envision how their functionality concepts could provide workers with visibility into whether handed off items have been received, reviewed, and acted on by intended parties.

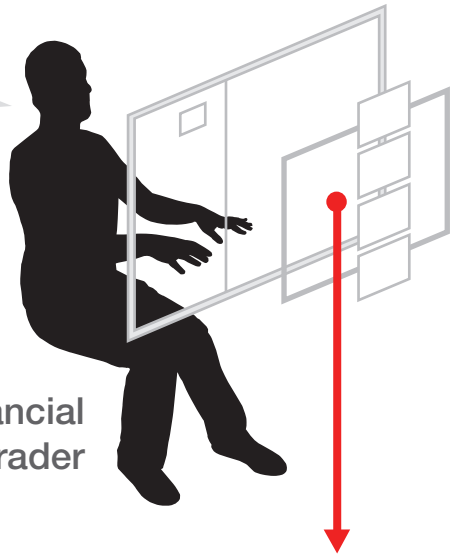
When product teams do not actively consider how explicit work handoffs could factor into their application concepts, resulting products may hinder the flow of work process with unclear transitions and undesirably vague rules surrounding individuals' responsibilities for valued work items (C9, G3).

Conversely, extra functionality for explicit work handoffs may not be necessary. For example, highly concrete work processes can exist as shared organizational norms outside of computing tools, reducing related technological needs.

See also: A, B, D5, H2, H3

I am getting too many trading messages at the moment...

I've got to delegate some of them in order to make sure that the work gets done fast enough...



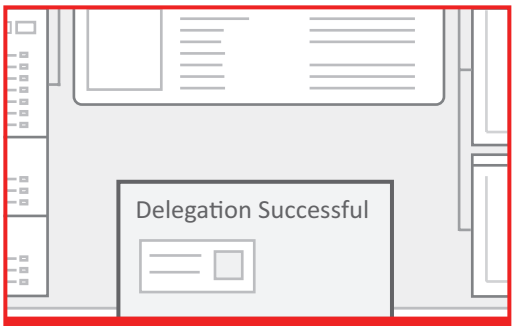
So I'm selecting some parts of my list...

The ones that I don't need to handle personally...

And now I'm sending those to the entire desk to see if anyone has the time to help me out...

I will holler in a minute if no one picks this up...

And, almost right away, I'm getting a message that Jon has just taken the list, so I don't have to think about it now...



Key *application envisioning* questions:

Where and when do handoffs occur in the knowledge work practices that your team is striving to mediate? What functionality concepts might your team envision to usefully support certain "special deliveries" of application content, closely tying them to sketched features for permissions and collaboration?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What role do handoffs currently play in targeted tasks and larger activities?

What, specifically, do targeted individuals hand off? What communication accompanies different types of handed off items?

How do specific types of handoffs fit within larger trajectories of work? Are they elements of defined work processes or improvised distributions of effort based on situational needs?

What breakdowns and errors can currently occur at handoff points? Could these problems represent potential opportunities for your product?

How might existing approaches for managing work handoffs be influenced by your team's application concepts?

In which work practices might your computing tool provide value by integrally supporting defined handoffs?

Outside of any structured delivery points, how might your team's sketched functionalities support workers' more open and emergent handoff choices?

What could the experience of handing off content be like? What feedback cues could allow senders to meaningfully know that their handoffs have been received, reviewed, or even acted on?

How might your team's approaches for supporting explicit work handoffs relate to your other functionality concepts for supporting cooperation, collaboration, and workspace awareness? For tailoring views of application content to particular identities?

How might handoff options be different when recipients are not users of your computing tool? How could "external" work remain clearly tied to your product?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J4. Authorship Awareness, Presence, and Contact Facilitation

Product teams can envision concepts for informative cues that could indicate who has worked, or is working, within a given functional area or on specific interaction objects. These cues can facilitate spontaneous communication between colleagues, both near and remote, and promote the traceability of distributed efforts.

Examples from three knowledge work domains:

> A scientist uses her laboratory's information management system to find out who accomplished several different experimental tasks on a particular clinical sample. She discovers which lab technician ran the sample through a certain instrument, and, seeing that he is currently online, she clicks on his name to launch a chat session, asking if he is free to talk **(see illustration on next page)**.

> An architect uses her building modeling application to look up which team members made a particular set of changes to a design. She sees that one of the change's authors is currently logged into the shared tool, so she walks over to have a conversation with him at his desk.

> A financial trader looks in his trading application to see whether any of his colleagues are currently trading at his firm's London office. If they are, he can simply click on their name to initiate a voice chat.

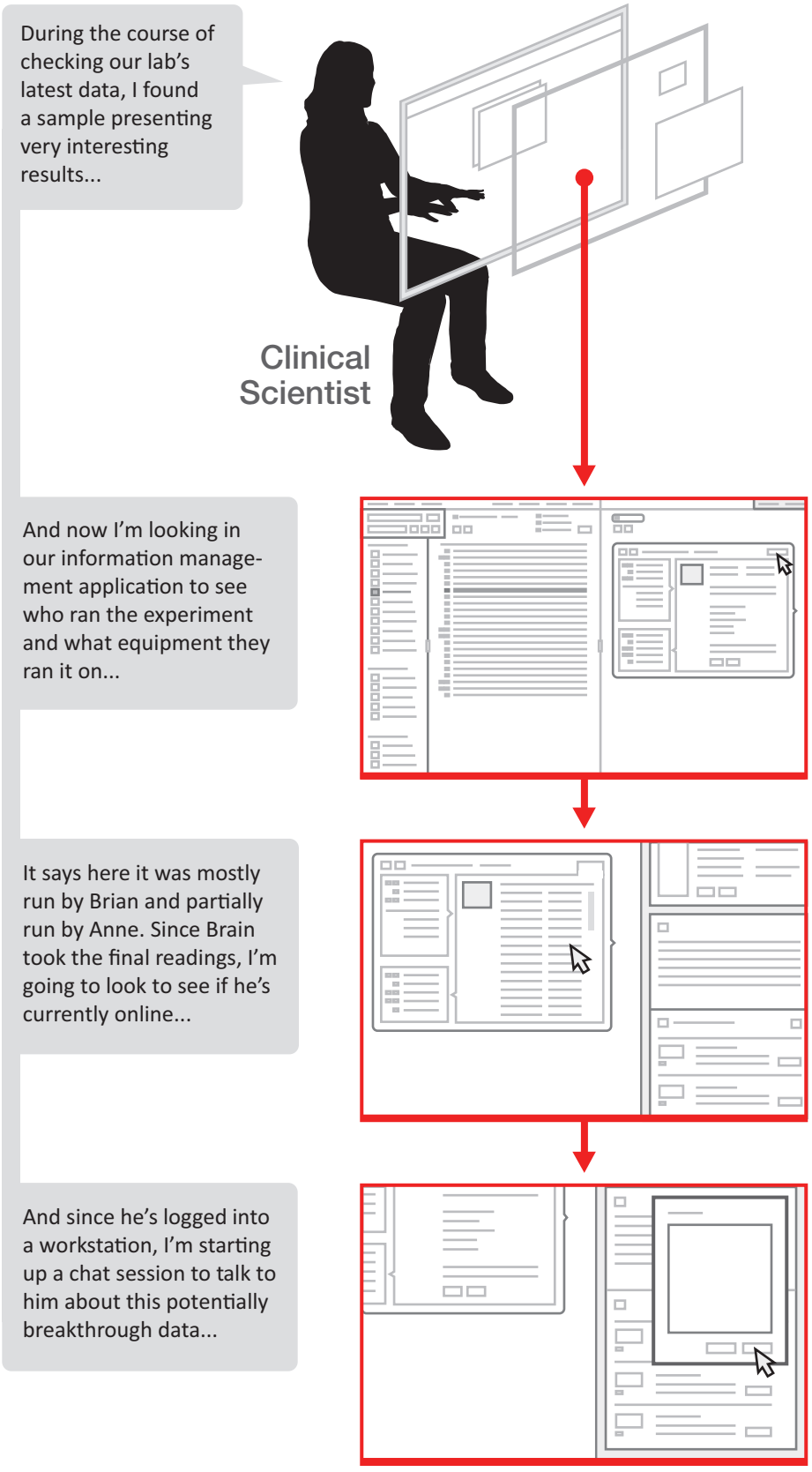
In offline practices, workers can often trace the author of a change through handwriting and other artifacts of how previous efforts were accomplished. Knowing if someone is available for conversation can mean looking across the room, going for a walk to another building location, or picking up the phone.

Product teams can envision a variety of functional responses that could create surrogates for physical cues, which may be lost when transitioning certain efforts and attentions to the screen. Teams can also consider how their computing environment might usefully enhance certain identity oriented possibilities. Automatically recorded trials of authorship data can be tied to specific interaction objects or functional areas (B, E3), becoming an essential part of their stored histories (H3) and enhancing workers' larger awareness of the actions of others within an application workspace (C7, G4). Authorship cues can also serve as a bridge for contacting relevant colleagues in order to clarify, extend, and question their work outcomes (L1).

When product teams do not actively consider the potential role of authorship and user presence cues in their application concepts, opportunities to promote effective communication and coordination can be lost. Workers may find it frustratingly difficult to discover who was responsible for particular changes (D2, D3), which may drive them to develop tedious and error prone work arounds (C9, G3, I7). And while contact facilitation can often be supported with separate, outside technologies, considering how this intent could be satisfied with integral functionality may allow teams to identify key, direct communication interactions in relation to their tools' valuable data (C4, D4, J1).

Conversely, too much visibility into the actions of others can be distracting (D4) and can potentially lead to unwanted surveillance effects (A2, G7).

See also: A, C5, D, E, F11


During the course of checking our lab's latest data, I found a sample presenting very interesting results...
Clinical Scientist
And now I'm looking in our information management application to see who ran the experiment and what equipment they ran it on...
It says here it was mostly run by Brian and partially run by Anne. Since Brain took the final readings, I'm going to look to see if he's currently online...
And since he's logged into a workstation, I'm starting up a chat session to talk to him about this potentially breakthrough data...

Key *application envisioning* questions:

With the goal of enhancing useful communication among users, how might your team's application concepts contextually present historical and real time cues about the "who" of others' actions and presence? How might targeted knowledge workers use these cues to initiate situated conversations?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What circumstances in the work practices that your team is striving to mediate currently lead people to investigate who has previously acted on an information artifact?

How do targeted individuals and their organizations currently keep track of authorship in various contexts?

What are the cultural norms, regulatory rules, and political implications around tracking worker's actions in targeted organizations? How is authorship information currently used in formal, procedural workflows and the evaluation of critical incidents?

Which tasks or larger activities currently involve impromptu, real time communications? Which are accomplished in relative seclusion?

How do targeted workers currently keep track of which colleagues are presently available for communication? How do they typically initiate conversations?

What larger design trends and advanced analogies to other domains could influence your team's ideas about thoughtfully highlighting authorship and presence information in your application concepts?

How might examining your sketched functional areas and interaction objects from the goal orientations of users' own collaborators help your team to envision different concepts for identity and presence cues?

How might "created by" and "modified by" attributes in interaction objects provide value in cooperative and collaborative work practices?

How might cues about individuals' actions or current presence be used as a means of initiating contact with them? What interruption effects may result on the receiver's end in these scenarios?

How might your team's approaches for supporting authorship, presence information, and contact facilitation relate to your other functionality concepts for supporting cooperation, collaboration, and workspace awareness?

What unwanted surveillance effects could unintentionally occur from strongly connecting users' identities and activities to specific application data?

What other privacy and security issues could be important to consider when envisioning functionalities that could be used to track workers' actions and lightweight, unstructured conversations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J5. Public Annotation

When workers make annotations in a specific context, they can direct their commentary to an intended audience, potentially reducing the difficulty of composing their communications. Product teams can envision concepts that could allow workers to annotate selected functional areas or interaction objects in ways that are visible and meaningful to desired recipients.

Examples from three knowledge work domains:

> An architect uses her building modeling application to work on early studies for the overall form of a new hospital. As she creates different versions of a central form idea, she inserts annotations of her design rationale in the models, knowing that these notes will be visible when she eventually presents the options to her colleagues, or if anyone happens to take the initiative to review the files on their own time **(see illustration on next page)**.
>
> A scientist uses her analysis application to review a number of data sets that have been recently generated in her clinical lab's ongoing experiments. As she explores the new results, she leaves comments about each data set so that when colleagues in her lab later review the study, they can agree or disagree with her interpretations.
>
> A financial trader, at the end of his work day, posts some thoughts on some recent, high value deals to a shared area of his trading application. He knows that colleagues in other global locations will likely read these notes when they first log in to start their day.

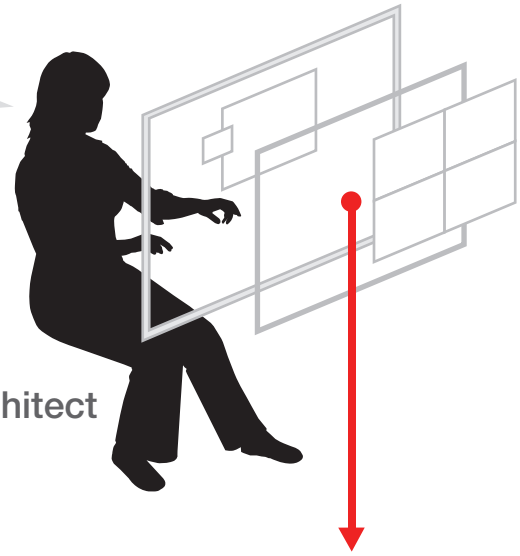
Successful knowledge work can rely on meaningful graffiti of a sort. Workers may place annotations in specific contexts, to be viewed by an anticipated "public" that will presumably interact with that location at a later time (C5). In some circumstances, these "markings" may only be valuable for a set duration. In other cases, contextual annotations can become essential elements of work artifacts, providing persistent, historical value in organizational memory (I7).

Although public annotations are often created for communication purposes, they can also serve as working annotations (H2) that offload workers' individual or collective memory effort (E1, E2), thereby supporting later reconstruction and cognitive tracing (H). Like working annotations, product teams can envision functional support for public annotation as textual notes, onscreen drawings, standardized categorical facets, attachments, links, and other means.

When product teams do not actively consider how public annotation could be supported in their application concepts (J1), resulting products may drive workers to use other media and communication channels in relation to certain onscreen content. These outside annotations may be difficult to coordinate with corresponding points within a computing tool (B1, B2, F1). The act of turning to outside media and channels can also be more effortful (D2, D3) than making a note directly where one is presently working. Lastly, real time communication used in place of public annotations may demand receivers' attentions at inappropriate times (D4).

See also: A, B6, J

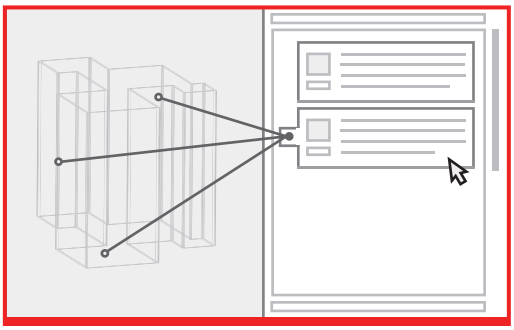
I'm trying out different rough forms for a new building that our firm is putting together a proposal for...



We don't have many requirements from the client, so as I create different model forms, I am typing up some comments that outline my rationale...

And I'm connecting the comment text to related areas within the draft building model...

And that way, when we are comparing different options, anyone on the team can use this view to read my ideas and justifications...



Key *application envisioning* questions:

Where, when, and how do knowledge workers currently annotate shared artifacts and environments in the work practices that your team is striving to meditate? How might targeted workers valuably communicate by annotating your product's functional areas and interaction objects with intended recipients in mind?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What workplace locations or objects in targeted organizations currently receive public annotations? What form can these annotations take?

What value do these public communications often provide within current work practices? Who are the intended audiences of specific kinds of annotations? Who else may view them?

What duration do various types of annotations have? Are they relatively static or are they iteratively placed and revised in a form of asynchronous conversation?

Which communication scenarios in your team's application concepts might be usefully supported through contextual notes and markings rather than interrupting, "separate" messages?

What methods of annotation could be appropriate, based on characterized communication needs, in your differing functionality concepts? Might textual notes, onscreen drawings, standardized categorical facets, attachments, or links be useful?

How could visual representations of public annotations contextually tie them to their onscreen subjects?

Who should be able to view whose notes, based on their permissions within your computing tool? How might workers select certain audiences for their annotations?

What useful supplemental interactivity and information might your team envision around various public annotations? For example, should workers be able to set durations after which their notes will fade from prominence?

How will collaborators discover that an annotation is present? Might contextual flags and synchronous messaging be useful to ensure that certain annotations are viewed by intended parties?

What supplemental attributes, such as a timestamp and authorship information, could be usefully included as part of public annotations?

What privacy and security issues could be important to consider when envisioning functionalities that would track workers' lightweight, unstructured conversations?

How might your team's approaches for supporting public annotation relate to your other functionality concepts for supporting cooperation, collaboration, and workspace awareness?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J6. Streamlined Standard Communications

Knowledge work often involves established, commonly shared genres of communication that play important roles in work activities and organizational memory. Product teams can envision functionality concepts that could provide workers with opportunities to offload some or all of the effort of creating, distributing, and interpreting these standard forms.

Examples from three knowledge work domains:

> A financial trader selects a deal in his trading application that he just made in error, then chooses an option to send a cancellation notice to his counterparty. The application generates a standard message with all of the necessary information to cancel the trade, and he adds a personal note to apologize for his miscalculation before sending it off **(see illustration on next page)**.
>
> An architect wants a drawing of a building's proposed north elevation to show her client. She instructs her building modeling application to automatically export a conventional elevation drawing based on a template that her studio has designed for this type of output.
>
> A scientist exports a canned report from her laboratory information management application. The compact, shareable document contains a set of standard data representations for a single clinical sample.

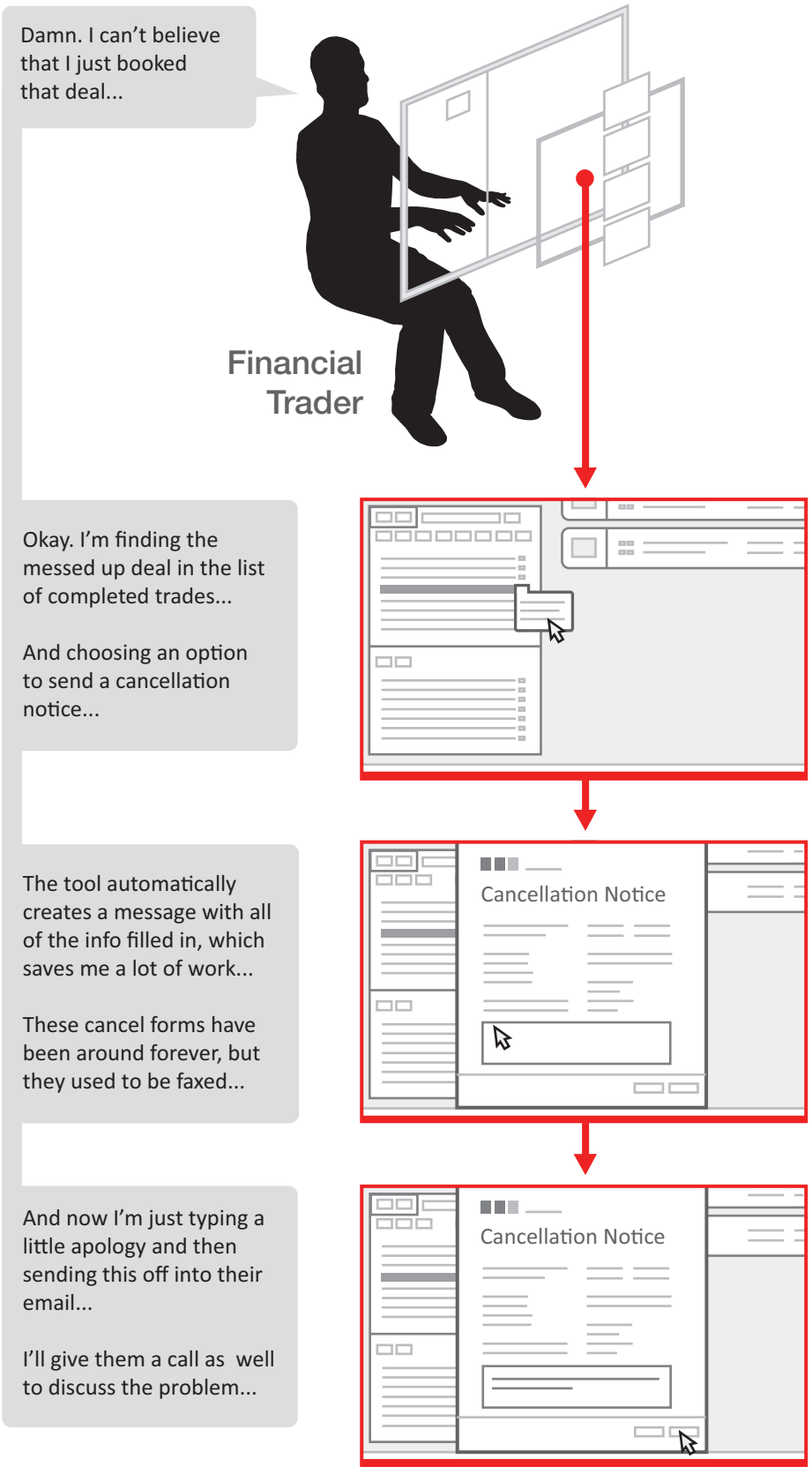
Knowledge workers often create conventional, expected discourse forms that can ease the communication burdens placed on both senders and receivers. From the perspective of individuals creating and distributing a communication, standard formats can scope the content of a fully formed message, shape the presentation of included content (F2), and define message recipients (A2, G7). From the receiver's perspective, conventional discourse forms can aid in interpretation and invoke specific understandings around a sender's intended purpose and meanings.

Depending on just how standard these established forms of communication are, product teams can envision functionality concepts that could valuably offload some or all of the effort (E3, E4) of creating, distributing (C4), and interpreting certain types of messages. There may also be opportunities to systematically improve the quality of specific message types through standardization (L1).

When product teams do not actively consider potential streamlining of standard communications in their application concepts, opportunities to support common discourse needs in knowledge work can be lost. When the intents behind these standard formats are not adequately supported, resulting applications can create communication fissures in targeted activities. Individuals and organizations may perceive these fissures as product deficiencies that overlook key opportunities to improve work efficiency and outcomes (D2, D3, K3).

Conversely, any standardization and automation of communication can carry certain risks and stifling drawbacks. Preformatted outputs may force the inclusion of some types of information, exclude needed categories of content, and constrain representation in unwanted ways (F, L1).

See also: A, B, C8, E, H1, H4, I1, I7, J


Damn. I can't believe that I just booked that deal...
Financial Trader
Okay. I'm finding the messed up deal in the list of completed trades...
And choosing an option to send a cancellation notice...
The tool automatically creates a message with all of the info filled in, which saves me a lot of work...
These cancel forms have been around forever, but they used to be faxed...
Cancellation Notice
And now I'm just typing a little apology and then sending this off into their email...
I'll give them a call as well to discuss the problem...
Cancellation Notice

Key *application envisioning* questions:

What standard communication formats are currently used in the knowledge work practices that your team is striving to mediate? What functionality concepts might your team envision to valuably automate and enhance the standardized portions of these communication tasks while still providing desirable levels of expressiveness and control?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How standard, in reality, are the conventional communications that are currently used within targeted organizations? Which are not very standard at all — from the rationalizing perspective of a product team defining a new computing tool?

How do targeted individuals use highly defined types of communication in their own operations, tasks, and larger activities?

What goals can trigger workers to create these communication forms? Who are they sent to? How frequent are different cases?

Do workers typically create these formal communications to ensure their persistence in organizational memory, or do people actually value how these formats can shape the scope and the content of their exchanges?

Who defined the standard formats that are currently in use? Have there been both top down and bottom up sources of standardization?

How have these schemes evolved into their present states over time?

What improvements might your team envision to enhance the usefulness and clarity of existing formats?

What functionality concepts might you sketch to support workers as they seek to offload effort that would otherwise be needed to create, distribute, and interpret high volume and consistently formatted communications?

Where in your team's models of work mediation might you identify new opportunities for valuable, largely automated standardization of communication outputs?

What flexibility might workers want in order to tailor standard communications to meet their local and situational goals? What options could allow them to informally annotate otherwise formal outputs?

Which contexts within your application concepts could valuably present clear and direct pathways for interactively creating defined communications?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

J7. Pervasive Printing

Many knowledge work tasks, including communication acts, can revolve around or be facilitated by paper documents. Product teams can envision functionality concepts that could allow workers to create various types of printouts while maintaining traceability back to their onscreen sources.

Examples from three knowledge work domains:

> A scientist prints a series of visualizations in her analysis application and passes them out as handouts in a laboratory meeting. The series of printed pages allows the group to collectively see a large clinical data set from a variety of perspectives. As they spot trends, she writes notes on her own set of printouts and dives into further visualizations on a shared projection screen **(see illustration on next page)**.

> An architect tries to work as much as feasible in her building modeling application so that her team can have access to her changes in something close to real time. However, there are still many parts of the design process, such as early ideation or the collaborative marking of quick corrections, where she finds it easier to sit at a table with her colleagues and communicate around printouts.

> A financial trader prints a problematic trade and hands it to a colleague. This interchange provides him with strong confirmation that the work, and all of the necessary information around it, has been handed off.

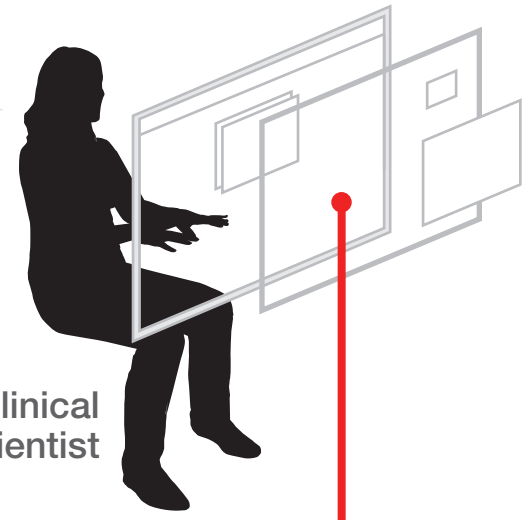
Repeated predictions of paperless futures, facilitated by computing in the workplace, have not come true. The reality of knowledge workers' observed practices often reveals that off screen representations of information, such as paper printouts, can afford many useful actions that are not yet commonly available in interactive computing. Because of these special affordances, workers may view printing functionality as a broad and pervasive necessity throughout their computing tools (A9).

Product teams can build upon their understandings of document usage in targeted work practices to envision potential printing functionalities within their application concepts. Standard printout formats can become essential and meaningful components in some tasks or larger activities (A, F). In communication acts, printed outputs can become transitory display media (H1) or formal outputs of work (J6, L1). People may appropriate printouts to support and track their explorations of potential outcomes (H), using paper "snapshots" to extend the effective areas of their computer displays through time. Individuals, groups, and organizations can also use paper records as a means of externally offloading memory efforts (E1, E2, I1, I7).

When product teams do not actively consider how knowledge workers might want to incorporate printing into their work practices, resulting applications may drive users to effortfully work around these limitations in order to make information available outside of their screens (D2, D3). In such cases, key benefits of accomplishing printing tasks from within a computing tool itself, such as formatting control or maintaining ties back into associated application content, may be lost (B3, F1).

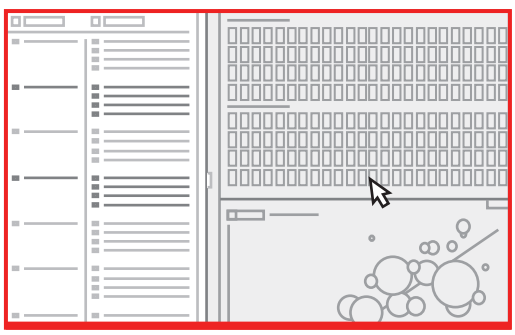
See also: J, G1, G7

I sometimes print out analysis work to share it in a lab meeting or to mark it up with hand written notes...



Clinical Scientist

PRINTOUTS SHARED
IN FACE TO FACE MEETING



And the notes that I take during our lab meeting discussions then feed back into my work in the analysis software...

Key *application envisioning* questions:

How do targeted knowledge workers currently use paper documents in the work practices that your team is striving to mediate? How might your team's application concepts allow workers to easily create valuable paper outputs of onscreen representations and content?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What affordances of paper documents do targeted individuals and organizations currently value?

How do off screen documents currently facilitate communication and collaboration?

How do targeted knowledge workers use paper instantiations of information to offload and distribute mental effort, such as short or long term memory burdens?

What role do printed records play in current approaches to progress tracking and archiving?

Which of your team's sketched interaction objects might be useful in printed form?

When could persistent printouts of larger views within your computing tool be useful in the context certain tasks or larger activities?

Where might targeted workers' off screen needs be common and frequent enough to provide tailored printing functionality for certain data perspectives?

What current printing needs could potentially be "solved" through onscreen interaction? Is the act of printing certain information currently a work around for clear deficiencies in current tools?

What general functionality concepts might your team envision to allow for selective printing of a broad range of application content types? What conventional design patterns and functionalities might you consider referencing?

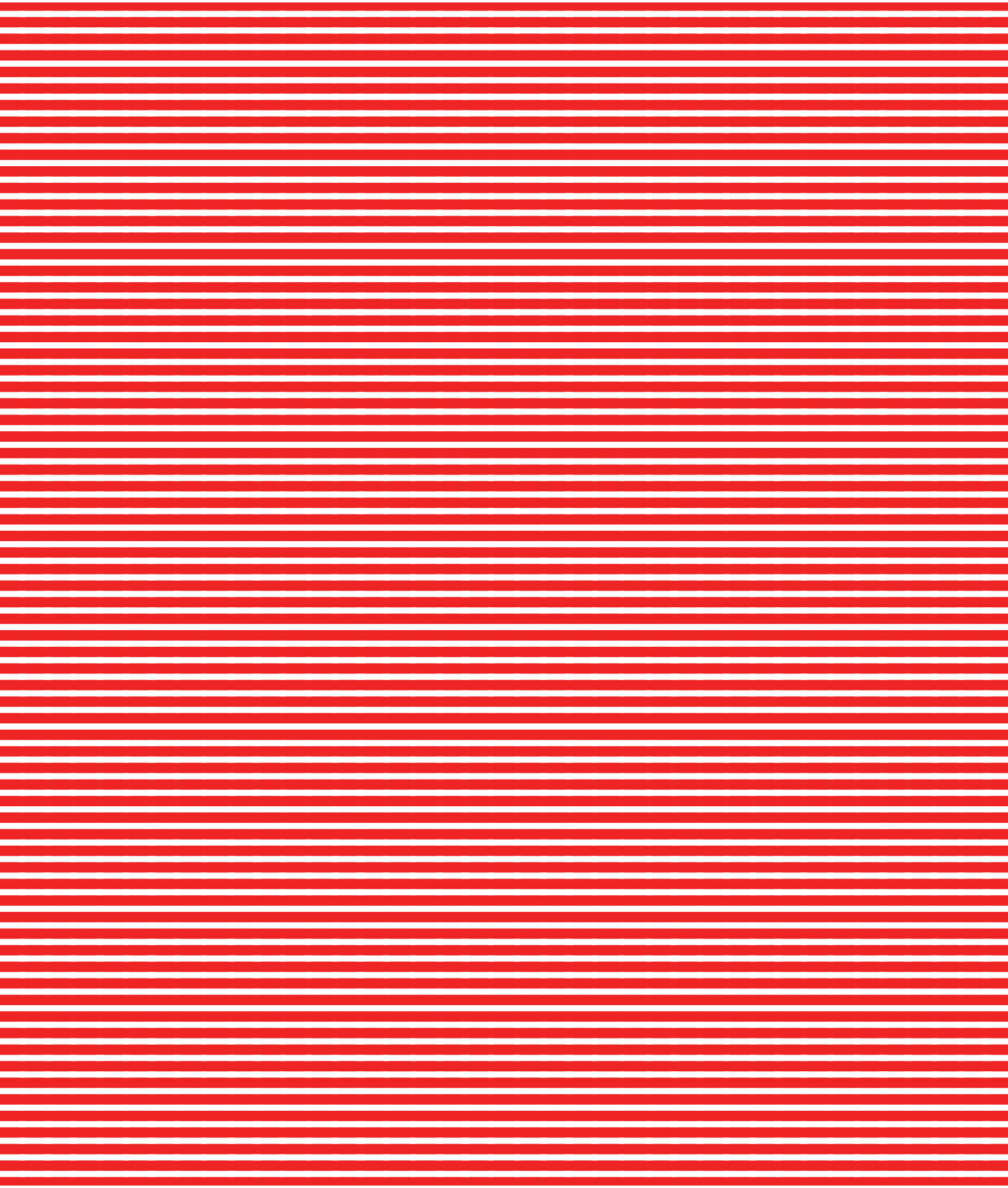
In what situations might it make sense for the information representations of printed outputs to be slightly different than how the same content is viewed onscreen?

What format improvements and supporting content might your team envision for the static medium of print?

What identifying information could help workers to tie the contents of a given printout back into your computing tool?

What formatting and content flexibility might workers want in order to tailor printouts to meet their local and situational goals?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?



K.

Promoting Integration into Work Practice

Valued computing tools can be designed to make "getting up to speed" as painless as possible.

Designing for such easy integration requires a clear understanding of the gaps that people will need to bridge in order to make use of a tool.

During *application envisioning*, product teams can map and explore how targeted knowledge workers and their organizations might integrate new onscreen offerings into their working cultures and technological systems.

By taking time to explore potential product adoption experiences — in an expansive sense — teams can identify opportunities to set the stage for direct, trusted, extensive, and meaningful use.

Although technologies can deterministically drive some changes once they are made part of working cultures, only individuals and organizations can truly determine whether computing tools will be successfully adopted into their own environments. Technologies do not create major cultural changes on their own, and brand messaging, or other background context, can only provide a frame for users' embedded, concrete, and personal experiences with a new product. That being said, the particular design characteristics of an application can play a major role in whether and how integration into practice occurs.

Rather than waiting for their technologies to be finished before thinking through potential adoption hurdles, product teams can consider adoption scenarios as part of generating their essential design strategy, envisioning services and functionality concepts to ease important learning and systemic challenges. Teams can envision these offerings and approaches as much broader responses than conventional, somewhat dissociated "user assistance," anticipating common needs and connecting with knowledge workers in meaningful and lasting ways.

This category contains 13 of the 100 *application envisioning* ideas in this book:

K1. Application localization

K2. Introductory user experience

K3. Recognizable applicability to targeted work

K4. Verification of operation

K5. Understanding and reframing alternate interpretations

K6. Design for frequency of access and skill acquisition

K7. Clear and comprehensive instructional assistance

K8. Seamless inter-application interactivity

K9. Directed application interoperation

K10. Openness to application integration and extension

K11. End user programming

K12. Trusted and credible processes and content

K13. Reliable and direct activity infrastructure

Product teams can use these ideas to explore specific means of supporting individual users, and larger customer organizations, as they transition from current practices to practices mediated by their new or updated computing tools. Early ideation and concepting focused on that support, rather than post hoc efforts during the final stages of a product's development, can help teams more fully integrate supportive options into their products' available user experiences.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A), "Considering workers' attentions" (D), and "Planning connection with use" (M) categories.

K1. Application Localization

Product teams can envision support in their application concepts for individuals from different cultural backgrounds. Targeted knowledge worker populations can have different wants and needs for the linguistic, symbolic, layout, and procedural aspects of their computing tools.

Examples from three knowledge work domains:

> An architect from Seattle is using a different computer during a visit to the Beijing office of her firm. She is pleased to learn that she can change the language from Mandarin to English in her own view of their shared building modeling application. This does not change the text of the informal notes that her Chinese colleagues have entered into design files, but it does allow her to navigate the application's interface labels in her native language **(see illustration on next page)**.
>
> A financial trader from New York, visiting the London office of his firm, finds that UK traders have set up their shared trading application to display their preferred spelling, time, and date standards. They have also added a number of UK specific fields in their standard trading forms that are not needed in the US domestic market.
>
> A scientist in San Francisco sends a clinical study file from her analysis application to a German colleague. She takes for granted that when her colleague opens it, the study file will be viewable in the localized, German language version of the software.

Many types of contemporary knowledge work are practiced in a number of global locations (A1), potentially driving a variety of application localization requirements. Even within a single customer organization, there can exist a diverse array of localization wants and needs.

Many product teams take for granted a certain linguistic and representational perspective when creating their computing tools. And, from within the influences of globalization, targeted workers may be familiar and somewhat comfortable with the experience of interpreting interfaces in a non native language (D2).

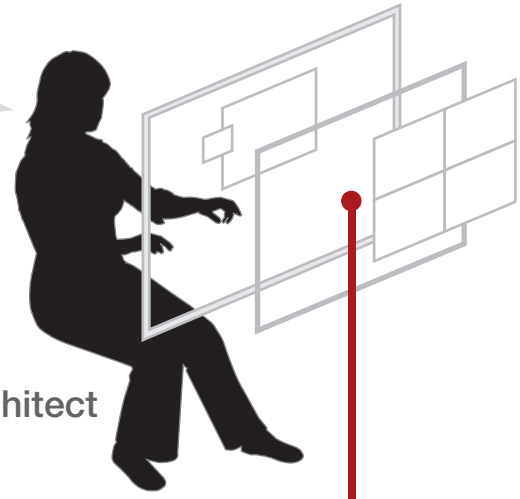
When a team does not wish to push their cultural frame onto their potential users, or cannot afford the impact of such a decision on their product's brand, they can envision localization approaches and functionality concepts based on targeted understandings of their intended audiences. While the languages available for display in an interface are often the most important localization factor (B1, C8), localization of symbolic content (F, K5), screen layout (C2, C4), and support for local variations in knowledge work practice (A7, A8) can also be crucial.

When product teams do not actively consider potential localization requirements for their application concepts, opportunities to be competitive in a broader range of markets (M4) may be lost. Knowledge workers in markets outside of a resulting product's originating region may face longer learning processes (D7, K2, K6) and a higher rate of errors (C9, G3). Additionally, teams may find it difficult to retrofit localization needs in a post hoc way due to their extensive, often structural, nature in interface design.

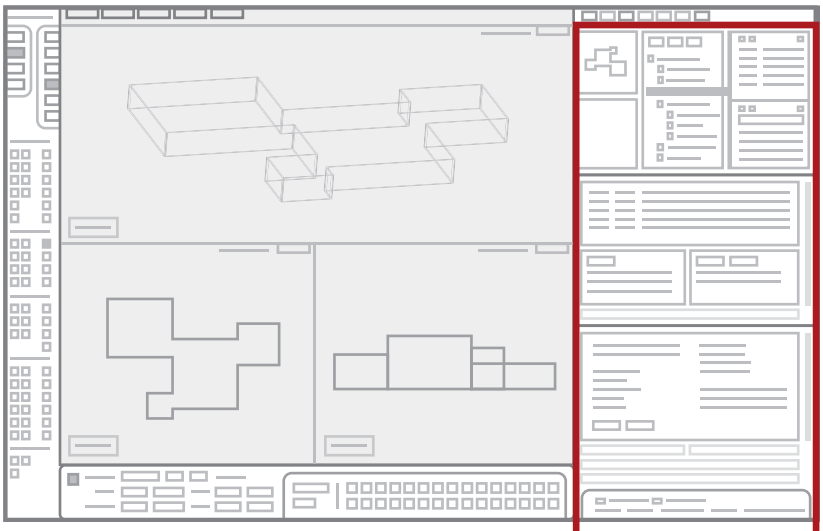
Conversely, intensive localization may negatively impact representational coordination and common ground between a product's audiences (F1, J2).

See also: A, C, G6, I, J6, K, L4, M

Sitting here in the Beijing office, the user interface of our firm's building modeling tool is in Mandarin...



So my colleague here told me how to change my language setting...



All user generated content in this panel remains untranslated from Mandarin

And now the application itself is in English...

But I will still need someone to translate all of the comments and info in this building model, because it was all typed in by this project team in the local language...

Key *application envisioning* questions:

In what localization intensive markets might your team be striving to provide a viable and desirable computing tool for knowledge work? What aspects of your application concepts could benefit from early envisioning around targeted local wants, needs, and opportunities?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What global markets might your team be targeting with your application, based on your emerging ideas about product strategy?

What impacts might the inclusion of various markets have on the sketched design strategies and brand positionings of your application concepts?

What separate linguistic audiences are likely to use your computing tool within and across these global regions?

What information does your team have about the specific localization needs within the breadth of your targeted audience segments? What do you not know?

How might this information impact your sketched directions for the linguistic, symbolic, layout, and procedural aspects of your application concepts?

What larger design and technology trends could influence your team's ideas about localization of your computing tool?

How might you prioritize localization needs in relation to other product design constraints? Which of these needs are strictly necessary? Which are desirable, "nice to haves"?

How could specific localization requirements influence the approach of your envisioned functionality concepts? Your sketches for high level application structures?

What unique opportunities could be present within your targeted locales? What design concepts might your team generate in order to explore these opportunities?

How might localization of shared views interfere with representational common ground in cooperative and collaborative work practices?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K2. Introductory User Experience

Product teams can envision how their application concepts could promote initial experiences that generate interest, instill confidence, clearly communicate essential information, and offer a direct foundation for committed adoption.

Examples from three knowledge work domains:

> A scientist launches her new analysis application for the first time and is presented with a set of interactive tutorials, each of which highlights one of several uses of the product. She selects an option that matches her clinical research goals and navigates through a well produced introductory tour **(see illustration on next page)**.
>
> An architect who has never used a building modeling application is anxious about making the switch from a more traditional CAD approach. The new tool presents a series of contextual training features, which allow her to learn about specific options at her own pace.
>
> A financial trader accesses a new version of his trading application and is presented with an option to view what has been updated in the latest release. From this quick, informative overview, he knows which new features he wants to try and which ongoing issues have been fixed.

Many contemporary computing tools for knowledge work do not excel at introducing themselves. Even when workers have been exposed to marketing materials, have decided that a brand and value proposition are compelling, and have acquired a new or improved tool, introductions are not yet complete until potential users have explored an application in their own activity contexts.

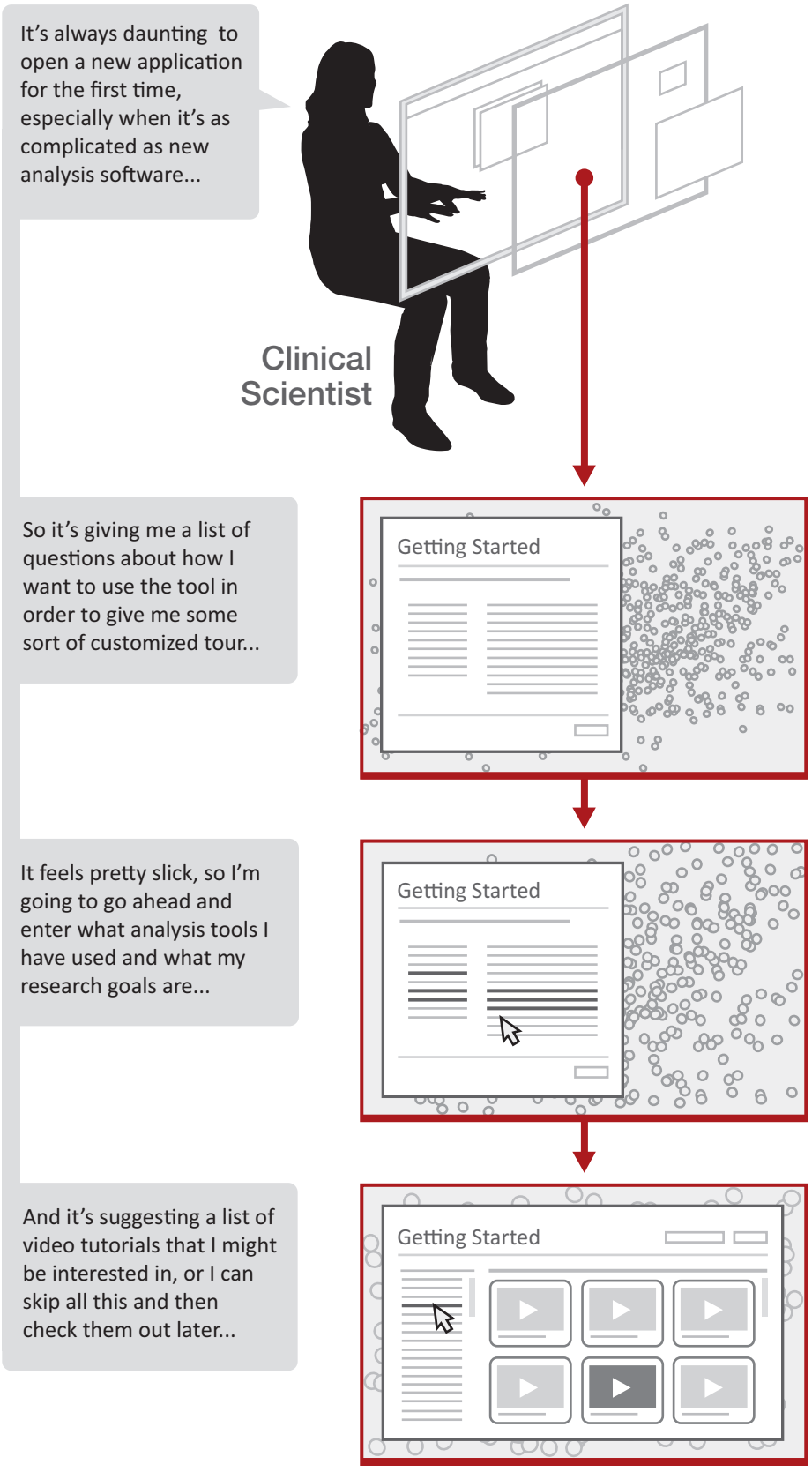
Individuals and organizations often do not have time to experiment with computing products (D3, K3). Knowledge workers may trust their early opinions about the desirability of an interactive application, especially when other product options are easily available.

Product teams can envision concepts for introductory functionalities that appear on the first occasion that a worker accesses an application or are spread across several early uses of a new tool. While application concepts with highly directive interaction models (C2) may require less introduction, products centered around open workspaces or novel conceptual models (C1, K5) may be understood faster and more completely with a scaffolding of initial instruction and suggested first steps (C4, G1, K7).

When product teams do not actively consider how their application concepts could drive meaningful early experiences, opportunities to make a product more desirable, to arm workers with useful understandings, and to prevent beginners' errors (C9, G3) can be lost. People may need to spend more effort determining what practices they can accomplish within an application (A, D2), how to verify its operation (K4), how to get started with their activities, and other key factors for making a computing tool integral to their own work.

Conversely, too much introduction can detract from direct engagement with the product functionality that workers will eventually use once they have chosen to fully adopt a tool.

See also: B1, C, F10, F11, H, J7, K, L4, M


It's always daunting to open a new application for the first time, especially when it's as complicated as new analysis software...
Clinical Scientist
So it's giving me a list of questions about how I want to use the tool in order to give me some sort of customized tour...
Getting Started
It feels pretty slick, so I'm going to go ahead and enter what analysis tools I have used and what my research goals are...
Getting Started
And it's suggesting a list of video tutorials that I might be interested in, or I can skip all this and then check them out later...
Getting Started

Key *application envisioning* questions:

Based on your team's understanding of targeted workers' current practices and background knowledge, what might they need to know in order to "get started" using your computing tool? What functionality concepts might your team envision to provide appropriate and dynamic instruction during these early user experiences?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What domain knowledge, existing skills, learned interaction expectations, and other background will targeted individuals likely bring to their early interactions with your team's product?

What big picture gaps might exist — at the overall level of your envisioned application concepts — between what workers already know and what they may need to know in order to have positive user experiences with your offerings?

What learning gaps might your team identify for each of your foundational functionality concepts?

What larger design trends and advanced analogies to other products and domains could influence your ideas about "out of the box" instructional experiences?

What conventional design patterns for early tutorials and topical "crash courses" might your team consider using?

What initial instruction concepts might you envision to provide an overview of your product's intended role in workers' practices?

What targeted concepts might you sketch with the goal of bridging specific, well characterized learning gaps?

What emotive and brand implications might your team focus on while exploring concepts for introductory experiences?

What media formats and visual approaches could appropriately represent certain types of instructional content in clear and engaging ways?

How might users' experiences with initial instruction offerings generate interest and instill confidence in your computing tool?

How might your concepts for initial instruction provide a foundation for, and potentially tie into, your other, more persistent user assistance options?

What interaction pathways could flow out of introductory experiences? How might users test new learnings through direct, constructive experimentation within your computing tool?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K3. Recognizable Applicability to Targeted Work

In order to communicate to potential users that the particulars of their work practices have been thoroughly considered, product teams can envision legible domain cues within their application concepts. When these cues are easily recognizable, knowledge workers may be more inclined to consider how they might use a new technology in their own activities.

Examples from three knowledge work domains:

> An architect likes that her new building modeling application uses language, conventions, and workflow that are specific to the practice of architecture. Other collaborative design products she has tried using in the past felt more like overly general 3D modeling tools that required her to excessively translate her ideas into arbitrary commands and design shapes **(see illustration on next page)**.
>
> A scientist finds that the features of her new analysis application are geared specifically to the types of clinical data explorations that she performs in her research. Whereas the general "data mining" tools her lab is migrating away from seemed arbitrary and unnecessarily difficult to learn, her new tool seems very approachable and relevant.
>
> As a financial trader quickly scans the menus and field names of the trial version of a new market information application, he recognizes standard options and functionalities that he is accustomed to using.

People judge how onscreen products could apply to their goals, while also considering any new opportunities that a application may facilitate. Computing tools that are intended for specific knowledge work practices can intentionally invoke their specializations through inherent branding and design. When targeted workers somehow see their own professional practices in a product's details, they may develop a focused interest in the tool, which can then evolve into committed adoption.

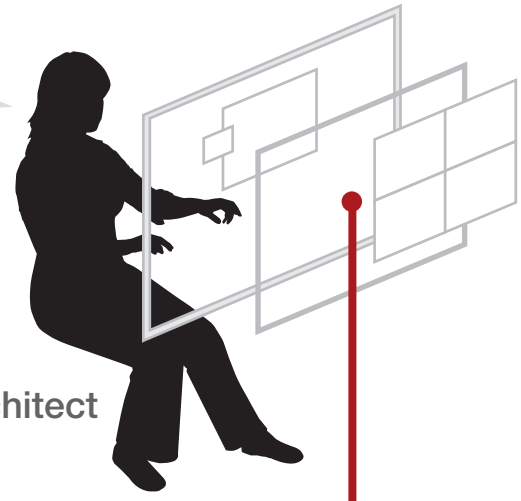
What this recognition may mean is highly contingent on the specifics of a given domain and the scope of work that a tool is intended to mediate. Given that expectations of applicability must be met with corresponding functional value, potential cues can include product genre (C1), interaction conventions (C2, L2), specialized language (B1) or information representations (F2), iconic design references (L3), and other domain specific elements (K2).

Product teams deliberately envisioning more generalized applications, to be used in multiple domains or markets, may face the challenge of not being able to leverage obvious references from any one specialty (A). Even without these literal cues, teams can uncover commonalities in work activities across specialties and then use these similarities to reveal legible, goal directed cues in their designs (B9, C4).

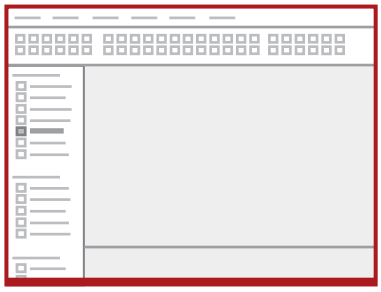
When product teams do not actively consider how they could make their application concepts a clearly recognizable part of the work that they are striving to mediate, resulting tools may fare poorly in workers' intuitive, "snap" judgments of utility. Depending on the market context, these judgments can impact both individual and organizational attitudes about acquiring and adopting products and brands (K).

See also: D1, F10, G1, L4, M

It's interesting to compare our firm's old building modeling tool to our new one...



PREVIOUS APPLICATION



CURRENT APPLICATION

"Side view"

"Rectangle tool"

"Surface Type"

"Engineering macro"

VS

"Elevation view"

"Define wall"

"Material: Exterior"

"Energy model"

Looking back at the software that we used to use, I honestly can't imagine using it again. It looks powerful, but it's very generic to 3d modeling, and my team had to work really hard to make it work for what we do...

Our current building modeling tool is completely built around the way we work. Just reading the labels and looking at the organization, it's all there...

Key *application envisioning* questions:

Beyond expected marketing messaging, how might the form, appearance, and behaviors of your team's computing tool rapidly communicate relevance for targeted knowledge workers' own goals and practices? What domain signs and emotive cues might workers feel a compelling affinity for while interacting with your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What meaningful consistencies in work practices across targeted organizations might your team translate into readily recognizable domain cues?

What clear commonalities in nomenclature and information representation could become visible references within your product?

Do your application concepts fit into — or somehow relate to — existing product genres that targeted individuals will likely know about? How might your team's design strategies play up these affinities while retaining meaningful brand differentiation?

What domain specific interaction patterns could trigger targeted knowledge workers to view your product as a potential addition to their technology environments?

How might the interactive entry points for primary pathways within your sketched computing tool provide a strong sense of domain relevancy?

How might design references to familiar and iconic artifacts improve potential users' "gut" judgments of your product's utility and applicability?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K4. Verification of Operation

Knowledge workers have specific understandings, within their organizations and communities of practice, of what it means to successfully accomplish their work. In order to support workers' ability to test whether their computing tools are operating as expected, product teams can envision functionality concepts around key verification scenarios.

Examples from three knowledge work domains:

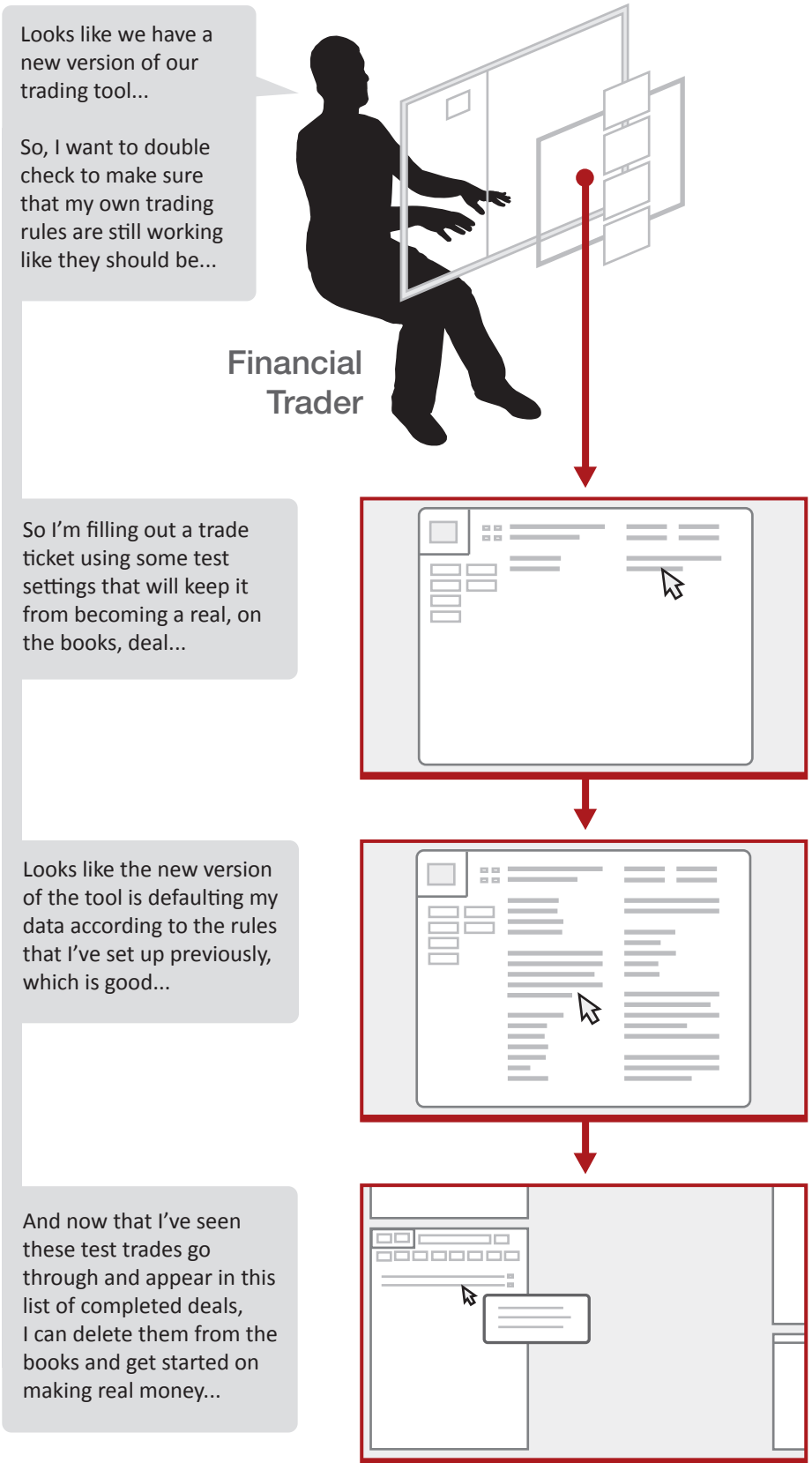
> A financial trader returns to work to find that a new version of his group's trading application has been installed. To test the installation, he makes some random trades between fake organizations, knowing that he can easily cancel these test deals once he is confident that the updated tool is working properly with the firm's many interconnected systems **(see illustration on next page)**.
>
> A scientist needs to make sure that a new data collection application provides the same experimental results as her lab's previous tool. To ensure that the results are comparable, she calibrates the new product and uses it to run test procedures on a set of clinical samples that have already been run using the previous tool.
>
> An architect completes a brief tutorial in her building modeling application, during which the tool runs checks to measure the performance of her current computing infrastructure.

Knowledge workers' early experiences of new technologies often involve critical minded testing. As part of adopting a new computing product, individuals and their organizations may need to see that a tool is functioning consistently and in line with its marketing claims. The more important the role of an application in work activities (A), the more emphasis may be placed on ensuring that it is operating as desired before putting it into use (B5, C7, C10, G4). Even after a new tool has been successfully tested within a workplace, individual workers may go so far as to run their own verification processes to ensure that their own high standards are being met (E5) and to gain a better understanding of how a product works (K2, K6).

What these verification processes could entail is heavily dependent on the roles that product teams are envisioning for their application concepts. To support potential testing scenarios in targeted work practices, teams can sketch guiding functionality concepts that could provide users with clear, instrumental outputs (F6). Definers and designers may actually mandate and automatically initiate some standard verifications within their products, such as testing certain features in the context of an organization's IT infrastructure (K10).

When product teams do not actively consider how individual workers and organizations will need or want to verify an application's successful operation, resulting computing tools may be difficult to adopt with the level of confidence that is often needed to support real world use (D2, D3). If workers' unguided, ad hoc verification processes produce mixed results, negative halo effects may lead people to abandon key functional areas or the entirety of a product or larger brand.

See also: C1, C9, F11, G1, G3, H, I, J, K, M


Looks like we have a new version of our trading tool...
So, I want to double check to make sure that my own trading rules are still working like they should be...
Financial Trader
So I'm filling out a trade ticket using some test settings that will keep it from becoming a real, on the books, deal...
Looks like the new version of the tool is defaulting my data according to the rules that I've set up previously, which is good...
And now that I've seen these test trades go through and appear in this list of completed deals, I can delete them from the books and get started on making real money...

Key *application envisioning* questions:

What mandatory or discretionary verification scenarios could be valuable for your team's application concepts? What aspects of your computing tool might targeted knowledge workers need or want to test in their local environments? What functionalities might you envision to directively enable certain well characterized checks?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Who is responsible for "officially" testing the functionality of new or updated applications within the organizations that your team is targeting?

What larger technology trends and advanced analogies to other domains could valuably inform your team's envisioning of product verification experiences?

Given the scope of knowledge work activities that your team is targeting, what types of operational checks might customer organizations require of your computing tool?

After an organization has tested your application, what verifications might targeted knowledge workers want to repeat themselves in order to confidently incorporate your new or updated product into their own work practices?

What operational verifications could be important to individual workers but may not seem as important to their organizations?

What tests might be beneficially rerun as your application is used through advancing versions of an organization's IT infrastructure over time?

How might your team envision the interactive procedures of infrequently accessed testing functionalities in a way that could allow users to easily and accurately run them without additional instruction?

What role could automation play in the flow of testing actions?

What streamlined instrumental displays might your team sketch with the goal of decisively presenting certain verification outcomes?

What potential errors or problems could arise in certain testing processes? How might they be prevented or handled in your sketched functionality concepts?

How might your ideas about application verification tie into your sketched directions for instructional assistance?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K5. Understanding and Reframing Alternate Interpretations

When product teams foresee potential "misinterpretations" of their functionality concepts — and these possibilities cannot be effectively "designed out" — they can envision cues that may help knowledge workers to reframe their own interpretations to be more closely aligned with their products' intended conceptual models.

Examples from three knowledge work domains:

> A scientist is used to thinking about certain areas of a scatter plot graph as representing outlier data within a set of clinical results. After she changes the chemistry that her lab uses to process samples, her analysis application provides some additional cues and instruction about how to interpret the new data **(see illustration on next page)**.
>
> An architect expects that all communications within her studio's building modeling application will be saved as part of the building model file. She is surprised to read that one type of communication is not saved, though the application's stated reason makes sense to her.
>
> A financial trader has used the same set of trading shortcut codes for years, but his group has recently decided to switch to a more efficient and all encompassing set. Now, when he accidentally enters an outdated shortcut code, his trading application suggests alternate codes that could match his intent.

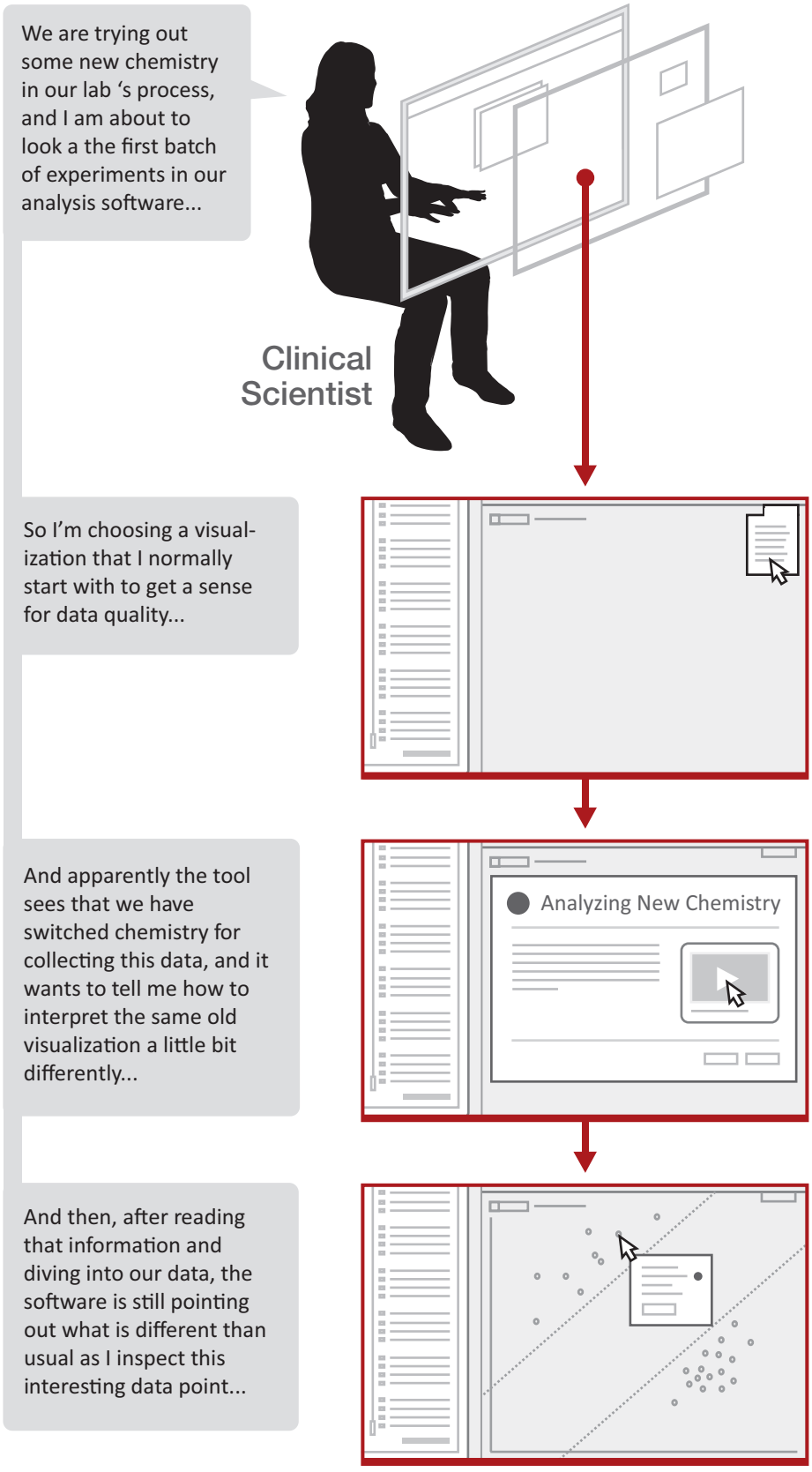
Some technologists talk of knowledge workers' "legacy" characteristics with dismay, as if trained professionals should simply abandon their cultures of practice for new processes that some believe to be more efficient or contemporary. Taking a more respectful approach, definers and designers can instead think of targeted workers' "legacy" of existing understandings and abilities as the core of valued skills that they are attempting to augment with their products.

Armed with the later perspective, product teams can recognize that valuable technologies often conform to, rather than rework, users' known practices. Teams can envision functionality concepts that harness knowledge workers' backgrounds, presenting them with useful new approaches in support of their existing working cultures (K2, K6, K7).

Even within this emphasis on intentionally suiting the design context, new technologies inevitably carry some novel ideas. By design, some of a product team's sketched concepts may necessarily operate in ways that can conflict with workers' existing understandings. Faced with these situations, teams can identify areas where damaging misinterpretations may occur and then envision ways to reframe these conflicts. These envisioning efforts can be particularly important when teams are seeking to deliver value by evolving or intentionally modifying some fundamental mental models within a knowledge work domain (D7, F2, F11).

When product teams do not actively consider potential alternate interpretations of their design concepts — and how potentially problematic interpretations could be reframed — resulting products may leave knowledge workers more susceptible to errors in decision making and action (C9, G3). Users may also find such applications to be difficult to learn, generally inefficient (D2, D3), and built on a flawed understanding of their motivations and needs (C1).

See also: A, G1, F, K, H, I4, L4, M


We are trying out some new chemistry in our lab 's process, and I am about to look a the first batch of experiments in our analysis software...
Clinical Scientist
So I'm choosing a visual-ization that I normally start with to get a sense for data quality...
Analyzing New Chemistry
And apparently the tool sees that we have switched chemistry for collecting this data, and it wants to tell me how to interpret the same old visualization a little bit differently...
And then, after reading that information and diving into our data, the software is still pointing out what is different than usual as I inspect this interesting data point...

Key *application envisioning* questions:

Where might targeted knowledge workers' domain background promote interpretations of your team's sketched computing tool that are different than those that you intended, potentially leading to errors and inefficiencies in use? What corrective cues and instruction might your functionality concepts include in order to reduce the likelihood of such conflicts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What domain knowledge, existing skills, learned interaction expectations, and other background will targeted individuals likely bring to their experiences with your team's product?

Where might peoples' existing understandings conflict with the conceptual models that your team is attempting to communicate throughout your sketched application concepts?

In which cases might it be better to redesign certain functionalities rather than attempting to reframe targeted workers' alternate interpretations of them?

Where might the opposite be true? Where could the value of your team's sketched approaches to mediating work be strong enough that you may want to try to respectfully mitigate and reframe users' alternate interpretations?

What larger design trends and advanced analogies to other products could influence your ideas about attempting to reframe certain conceptions within your computing tool?

What corrective cues, instructional content, and other design communication could reduce the incidence of potential misinterpretations?

How might your concepts for reframing alternate interpretations tie into other instructional assistance approaches in your application concepts? How might they relate to your approaches for error prevention and handling?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K6. Design for Frequency of Access and Skill Acquisition

Knowledge workers become highly familiar with some parts of their interactive applications and remain "perpetual intermediates" or even novices in others. Product teams can envision appropriate levels of interaction constraint and instruction for different functionality concepts, matching design responses to expected frequency of use.

Examples from three knowledge work domains:

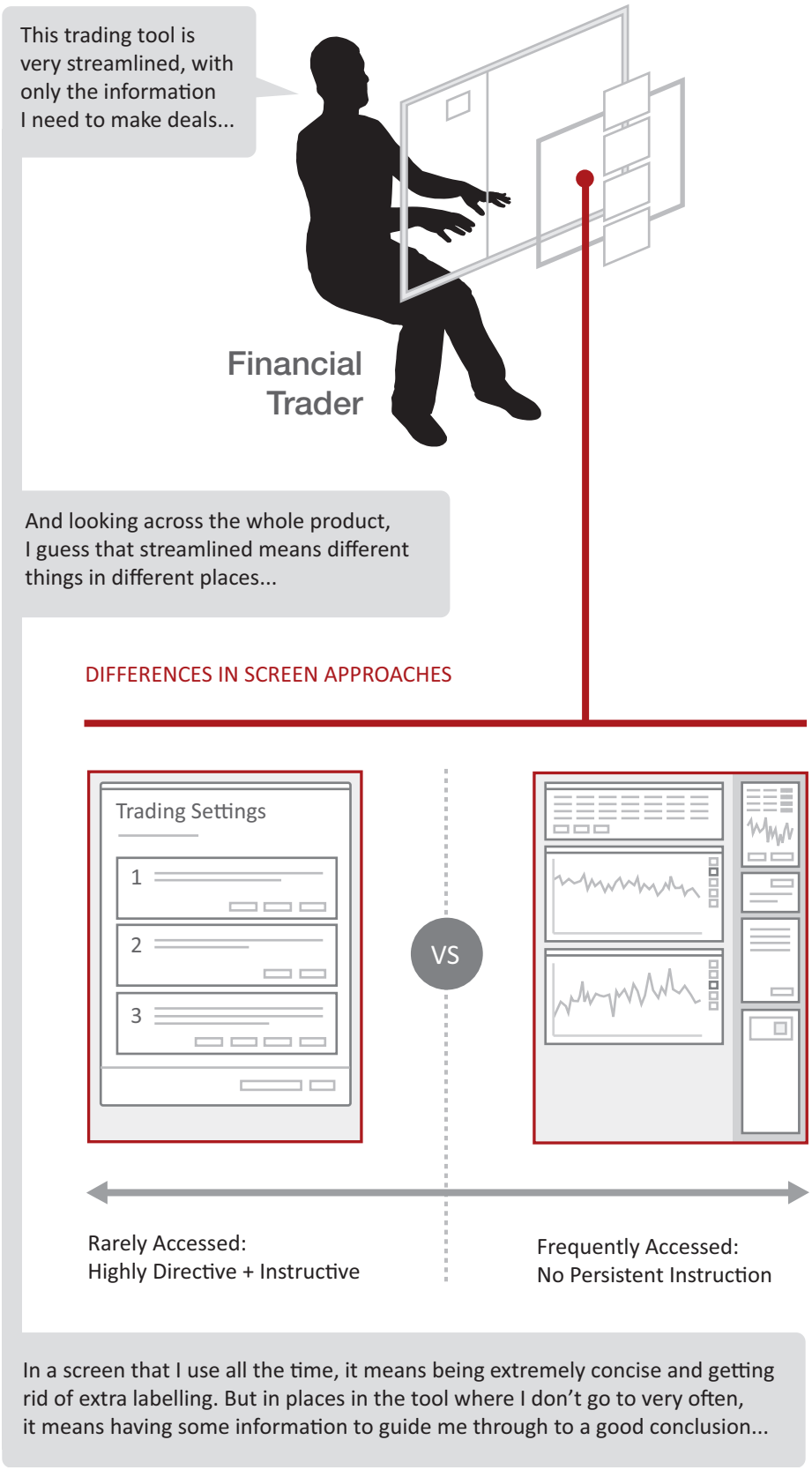
> A financial trader likes that the trading application screens that he uses throughout his day contain only essential content and functionality, without any instructions or extras. When he accesses settings dialogs and other secondary areas of the computing tool, he often reads contextual instructions without realizing that he is using them **(see illustration on next page)**.
>
> An architect has learned specialized shortcuts to rapidly interact with her building modeling application without having to shift her focus to its menu structure. Since she uses certain functions very frequently, she had no qualms about learning these somewhat arbitrary interaction mappings.
>
> A scientist mostly just wants to see data visualizations on her screen, not excess interface controls or content, but she likes to have quick access to key definitions. Each of the tools that she uses does things slightly differently and she sometimes forgets what some of the data parameters in a given screen actually mean.

Even when a knowledge worker "knows how" to use a computing tool, they typically do not remember how to use every one of its functions. Additionally, the idea of "knowing how" to use a given functional area can mean very different things — a single interactive application can contain some functionalities that are as complicated to learn as a musical instrument and other areas that are as directive and restrictive as an automated teller machine (C3). In either case, people learn through their ongoing experiences, though the investment involved and character of their resulting skills can differ greatly.

Based on characterizations of use (A, D2), product teams can envision different learnability requirements for their various functionality concepts. For frequently used functionalities, very little aid may be needed outside of introductory experiences (K2, D7). For infrequently accessed functional options, computing tools can attempt to scaffold workers based on their assumed goals and knowledge (C1, K5), as well as targeted requirements for flexibility and error prevention (C9, G3).

When product teams do not actively consider how different functional areas within their application concepts will be differentially accessed and used, resulting products may convey an inherent disregard for learning experiences (D3). Teams may inappropriately treat all areas of a product as if they will be self explanatory (H), supported by separate or entirely distant instructional content (K7). Alternately, poorly envisioned applications can become overly directive and instructive in primary areas where people are likely to eventually find such scaffolding to be distracting (A9, C6, D4).

See also: C, E, G, K, M


This trading tool is very streamlined, with only the information I need to make deals...
Financial Trader
And looking across the whole product, I guess that streamlined means different things in different places...
DIFFERENCES IN SCREEN APPROACHES
Trading Settings
1
2
3
VS
Rarely Accessed:
Highly Directive + Instructive
Frequently Accessed:
No Persistent Instruction
In a screen that I use all the time, it means being extremely concise and getting rid of extra labelling. But in places in the tool where I don't go to very often, it means having some information to guide me through to a good conclusion...

Key *application envisioning* questions:

How might your team characterize predicted frequency of use for each of your sketched functionality concepts? How might these differential levels of access, along with other relevant learnability factors, impact the amount of direction and scaffolding that you incorporate into each interaction pathway?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How much learning investment might targeted individuals be willing to make in order to use the various functionality concepts that your team has envisioned?

How might workers' expectations around skill acquisition vary based on the value that they assign to a given functional option in the context of their own work practices?

How might your product's overall strategic message and brand promise affect users' motivations? What sources of value could be compelling enough to seem worthy of learning effort?

What areas and pathways in your team's application concepts are likely to be accessed frequently and will probably become well known through normal use?

What portions of your computing tool will users, by design, rarely access?

What gap exists between what targeted workers already know and what they may need to know in order to have positive experiences with your sketched functionalities?

How might your team use the above understandings to categorically prioritize tradeoffs between initial learnability and skilled use in different parts of your product?

In particular, how might your team place an emphasis on envisioning appropriate interaction constraints and instructions for complex functional areas that workers will rarely see?

Where might directive limits on interactive flexibility valuably steer users toward their goals and prevent some types of errors?

Where might contextual instruction provide value in different functionality concepts? How might your team present this instructional content in clear and engaging ways?

Where might it make sense to drive users toward an application's comprehensive instructional assistance offerings, rather than include such content in the context of "normal" interactions?

How might your envisioned approach for supporting different types of skill acquisition tie into your concepts for introductory experiences? How might it relate to your conventions for error prevention and handling?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K7. Clear and Comprehensive Instructional Assistance

The balancing act between initial learnability and long term usability often results in some functionalities that are not self explanatory to all knowledge workers in a targeted population. To ensure that workers have just-in-time access to needed answers, product teams can envision useful, findable, and directive "help," delivered via channels that are well suited to characterized learning needs.

Examples from three knowledge work domains:

> A scientist sees an interesting trend in her analysis application, and she wants to modify her current visualization to highlight data that match some complex criteria. After clicking through some settings screens without success, she clicks on a contextual help icon to launch the relevant section of the application's comprehensive help system **(see illustration on next page)**.
>
> An architect browses the support website of her building modeling application for "tips and tricks" on how to set up security permissions for subcontractors, before getting started on the necessary data entry.
>
> A financial trader uses the phone number listed in an error message to call his trading application's customer support team, whose members have access to an extensive help database.

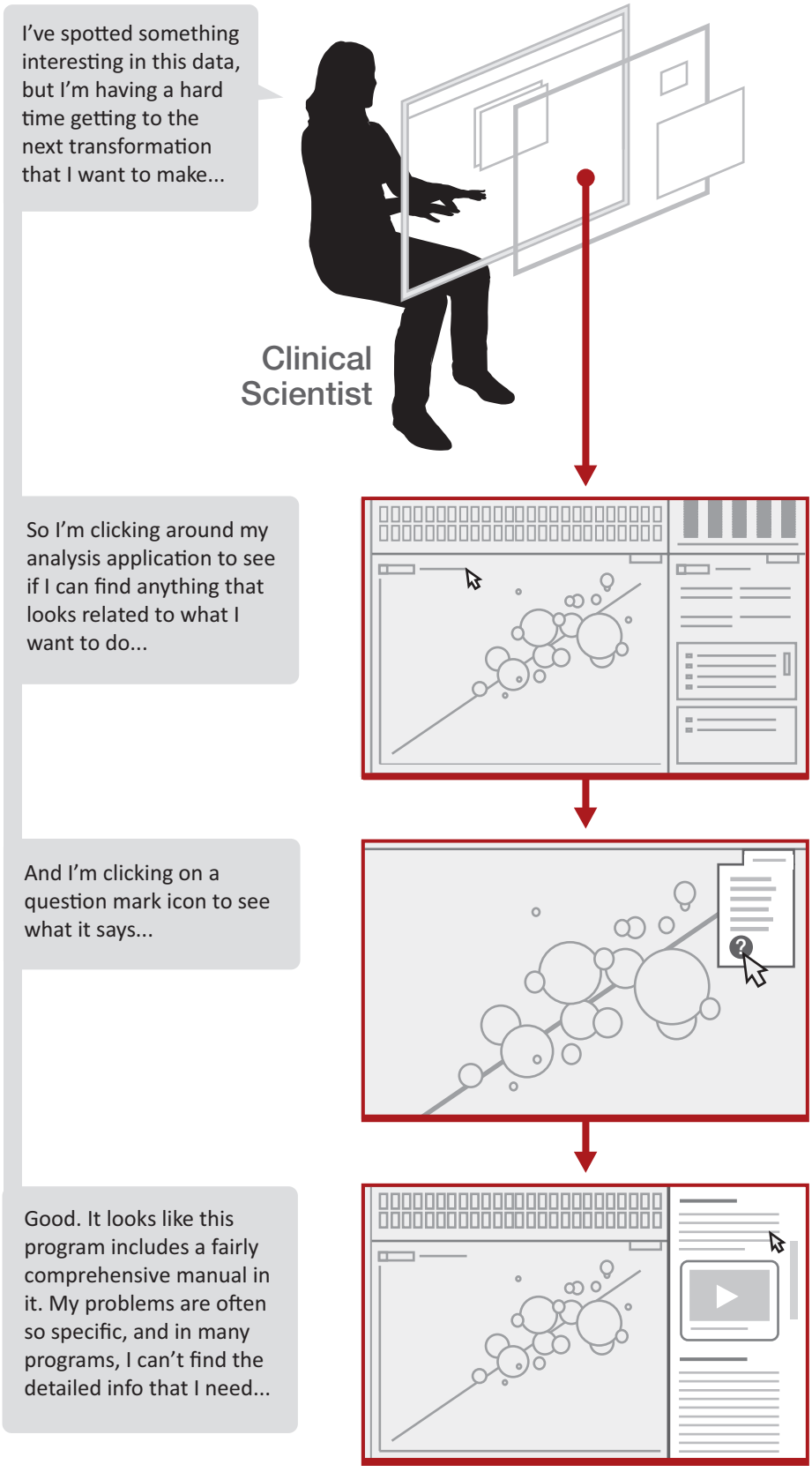
Knowledge workers often do not learn an entire interactive application in a single sitting, and supporting instruction can play a crucial role as the adoption process unfolds over time in organizations. Computing tools for specialized work can be extremely specific and intricate, making it difficult for people to digest all of the instruction that they need during initial training sessions or "out of box" experiences (K2).

Product teams can envision a variety of instructional methods and presentations within their application concepts that are tailored to the range of ideas that they are seeking to communicate (C3, F, L4). In some instances, distinct assistance options, somewhat removed from day to day user interfaces, can be desirable. For example, contextual instruction (D4, F1) can progressively disclose an application's help content, an online information repository, the social networks of a user community (I, M3), or a direct communication channel to product support representatives. In other cases, where actions are infrequently accomplished, procedurally complex, or especially sensitive to error (G3), workers may value scaffolding instruction that is more present and integrated into user interfaces (K6).

When product teams do not actively consider the larger instructional assistance requirements of their application concepts, resulting products may not adequately support workers' initial and ongoing learning needs (K5, C1). These tools may present users and their organizations with costly and frustrating trial and error situations (D2, D3) that can negatively impact brand perceptions and work outcomes (L1).

Conversely, poorly conceived instructional features may provide little value. Many contemporary tools contain vague, marginally useful instructional content of the baffling sort that most personal computer users are all too familiar with.

See also: A, C, E, F10, F11, G, J7, K, M


I've spotted something interesting in this data, but I'm having a hard time getting to the next transformation that I want to make...
Clinical Scientist
So I'm clicking around my analysis application to see if I can find anything that looks related to what I want to do...
And I'm clicking on a question mark icon to see what it says...
Good. It looks like this program includes a fairly comprehensive manual in it. My problems are often so specific, and in many programs, I can't find the detailed info that I need...

Key *application envisioning* questions:

What functionality concepts might your team envision to provide targeted knowledge workers with comprehensive and appropriate support for their learning needs and critical issues? What contextual, goal directed interaction pathways could your computing tool present in order to connect users with stored user assistance content, online repositories, relevant social networks, or specialized support staff?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What domain knowledge, existing skills, learned interaction expectations, and other background will targeted individuals likely bring to their experiences with your team's product?

What parts of your application concepts may be inherently difficult for some workers to learn? Where might variabilities in work practices within targeted markets and organizations lead to additional learning needs for some users?

What high level gaps exist between what workers already know and what they may need to know in order to have positive user experiences with your computing tool?

What specific understanding gaps might your team identify for each of your primary functionality concepts?

What portions of your application concepts will probably be accessed infrequently in most organizations? What learning needs could arise from these cases?

Which of your sketched functional areas has, by design, more flexibility and less instructional content, with the assumption that workers will gain skills through ongoing use and would find directive scaffolding to be distracting?

What comprehensive assistance approaches are most appropriate for the learning needs your team has characterized?

In what cases might actual conversations be necessary to resolve knowledge workers' issues? What feasible support can your team envision for these interactions?

How could the availability of instructional support be made contextually apparent in your computing tool without distracting from rehearsed, day to day interactions?

What media formats could appropriately represent canned instructional content in clear and engaging ways?

How might your envisioned approach for providing comprehensive instructional assistance tie into your concepts for introductory instruction? How might it relate to your conventions for error prevention and handling?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K8. Seamless Inter-application Interactivity

Knowledge workers may need to interact with several computing tools in order to accomplish their activities, effectively treating their adopted suite of applications as one overall system. Product teams can envision functionality concepts that could facilitate desirable and fluid onscreen interactions across related products.

Examples from three knowledge work domains:

> An architect selects and copies a section of a building model, then pastes it into an email that she will send to an acoustical consultant who is estimating some specific costs. When the consultant receives the file, he is able to open it in a different 3D modeling tool, where only the specific section of the building under discussion appears on his screen **(see illustration on next page)**.
>
> A financial trader copies content from a series of forms in his trading application and pastes it into a spreadsheet for further analysis. Since he knows his spreadsheet application very well, he often prefers to work this way, even when other products provide "spreadsheet like" functionality.
>
> A scientist selects several rows in a table within her analysis application and drags them into a presentation document. The dragged content then appears in the body of the document with the same table formatting.

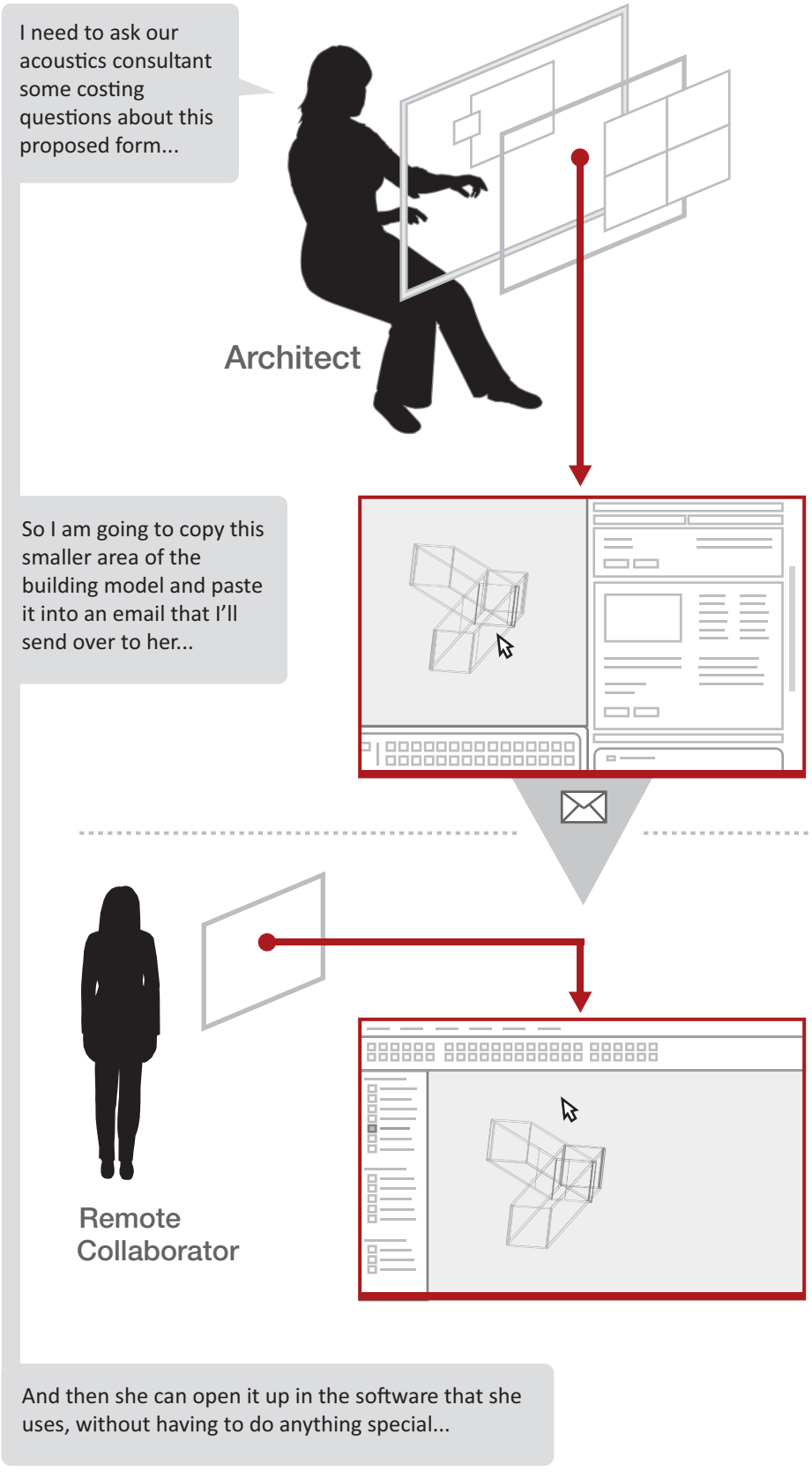
In many knowledge workplaces, individuals' screens are frequently alight with several different applications at the same time, often for overlapping purposes. An individual product is often only one component of an overall system of tools that workers appropriate to accomplish their activities.

Based on analyses of common product and activity interrelations in targeted work practices (A5), product teams can envision functionality concepts that could provide users with lightweight and tightly coupled opportunities to effectively tie into their other computing tools. Conventional examples of these interoperations include "cut and paste" and "drag and drop" of interaction objects that users may want to move from one product to another (B1, B8, E3, G2).

When product teams do not actively consider how knowledge workers may want to seamlessly transfer content into and out of their applications, resulting products can contain functionally "isolating" barriers to long term productivity and satisfaction (D2, D3, M4). Workers may traverse issues that could be solved with lightweight interaction by redundantly entering data, capturing screens, printing information (J7), or exporting and importing application content (K9). Even with available workarounds, deficiencies in seamless interactivity may become an early and frequent complaint about adopted products (D4, G3, M4).

Conversely, for reasons involving top down business or brand strategy, teams may intentionally decide to keep their application concepts closed to this sort of interoperation, regardless of the value it could deliver to workers and their organizations. However, over time — and under the influence of Internet driven thinking — this line of "protective" reasoning appears to be becoming less prevalent in many workplace computing domains.

See also: A, B, C3, E, G, I5, K, M


I need to ask our acoustics consultant some costing questions about this proposed form...
Architect
So I am going to copy this smaller area of the building model and paste it into an email that I'll send over to her...
Remote Collaborator
And then she can open it up in the software that she uses, without having to do anything special...

Key *application envisioning* questions:

Which of the work practices that your team is striving to mediate could span multiple computing tools in knowledge workers' technology environments? What useful interactions might your team envision to allow targeted workers to dynamically use multiple onscreen applications as if they were a single seamless system?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which computing tools do targeted individuals primarily use, given the larger constellation of technologies that are available to them?

What role do each of these tools play within your team's targeted tasks and larger activities?

How do knowledge workers currently coordinate their various onscreen applications in order to accomplish their goals in different scenarios?

Which interoperations frequently result from lightweight, spur of the moment choices?

What breakdowns and errors can occur in these interoperations? Could these problems represent potential opportunities for your product?

Where might your team's sketched strategic directions suggest "open" and networked approaches to other technologies in targeted workers' environments? Where might they suggest "closed" approaches?

What market trends and technological realities might your team consider while envisioning possibilities for dynamic interactions between specific computing tools?

What attitudes and expectations do targeted individuals and their organizations have regarding seamless interactivity as a means of bridging their various applications?

Which of your sketched interaction objects and functionality concepts might provide value in the context of other computing tools?

What interaction objects from other computing tools might provide value in the context of your team's application concepts?

Where might existing convections for lightweight inter-application interactivity, such as "cut and paste" and "drag and drop," play a role in your functionality concepts?

What novel interaction approaches might your team envision for specific bridging operations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K9. Directed Application Interoperation

Knowledge workers may want to accomplish their activities by using a series of functionalities that sequentially span more than one computing application. To allow for the movement of large volumes of data in relevant formats, product teams can envision functionality concepts that could facilitate cross boundary interoperations with distinct import and export options.

Examples from three knowledge work domains:

> A scientist exports a set of clinical data from her new analysis application so that she can import it into her old analysis tool, which includes some different visualizations. Depending on what she discovers in the older tool, she will likely import a subset of the clinical data back into her new, primary tool for further examination **(see illustration on next page)**.
>
> A financial trader, who wants to better understand the potential long term value of a large trade offer, exports the offer's data into a format that he can then easily uploaded into his preferred market information tool.
>
> An architect needs to post some plans on an extranet website for her clients to view. Since these clients do not have the specialized software applications that her studio uses, she instructs her building modeling application to export the selected plans as web pages.

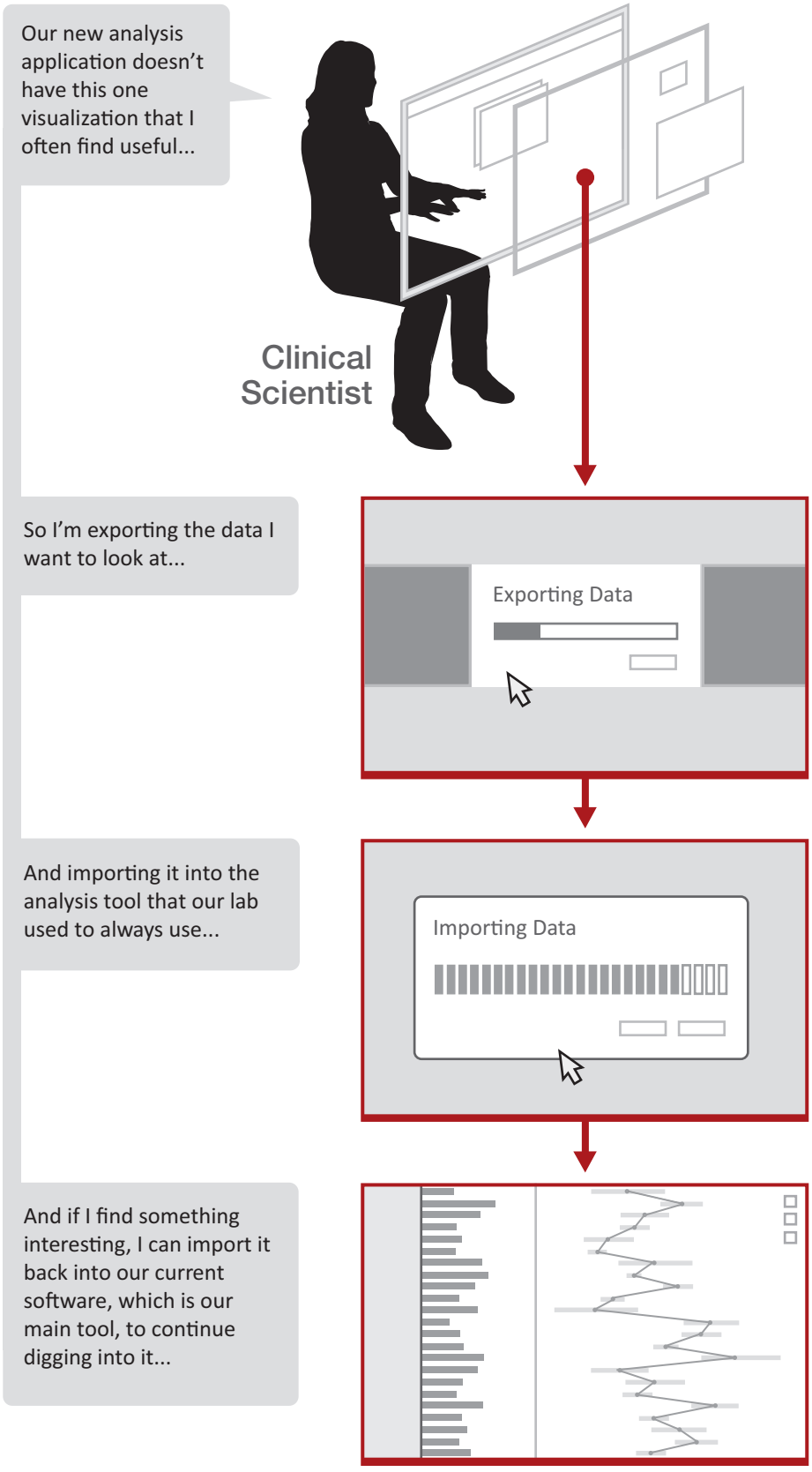
When dealing with complex information, large volumes of data, and entire arcs of activity, knowledge workers frequently want or need to apply several different computing tools to their efforts. Individual products are often only one component of an overall system of tools that workers have appropriated to accomplish their activities.

Based on analyses of common product and activity interrelations in targeted work practices (A5), product teams can provide workers with opportunities to sequentially bridge applications through the export of their product's content and the import of related content from other tools. Desirable approaches and formats for these functionality concepts can be highly contingent on teams' predictions about how their product will be used in conjunction with other technologies. Support for more open standards may promote directed interoperation with a variety of computing tools, including those that may not yet exist during *application envisioning* or at the time of a product's eventual release (M4).

When product teams do not actively consider how knowledge workers may want to directly interoperate their applications with other on-screen tools, resulting products may be experienced as inaccessible "islands" of functionality. These applications may pose critical barriers to long term productivity and satisfaction (D2, D3, G3), creating a need for redundant data entry (E3, E4) that can take the place of higher order, and potentially higher value, pursuits (D4).

Conversely, for reasons involving top down business or brand strategy, teams may intentionally decide to keep their application concepts closed to this sort of interoperation, regardless of the value it could deliver to workers and their organizations. However, over time — and under the influence of Internet driven thinking — this line of "protective" reasoning appears to be becoming less prevalent in many workplace computing domains.

See also: A, B, C8, E, G1, I5, K, M


Our new analysis application doesn't have this one visualization that I often find useful...
Clinical Scientist
So I'm exporting the data I want to look at...
Exporting Data
And importing it into the analysis tool that our lab used to always use...
Importing Data
And if I find something interesting, I can import it back into our current software, which is our main tool, to continue digging into it...

Key *application envisioning* questions:

Which of the work practices that your team is striving to mediate could bridge multiple computing tools in knowledge workers' technology environments? What separate, named functionality concepts might your team envision to allow targeted workers to valuably move selected collections of application content across otherwise isolating product boundaries?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which computing tools do targeted individuals primarily use, given the larger constellation of technologies that are available to them?

What role do each of these tools play within your team's targeted tasks and larger activities?

How do knowledge workers currently coordinate their various onscreen applications to accomplish their goals in different scenarios?

Which interoperations frequently result from actively and sequentially moving data between computing tools?

What breakdowns and errors can occur in these interoperations? Could these problems represent potential opportunities for your product?

Where might your team's sketched strategic directions suggest "open" and networked approaches to other technologies in targeted workers' environments? Where might they suggest "closed" approaches?

What market trends and technological realities might your team consider while envisioning possibilities for the directed movement of data between certain computing tools?

What attitudes and expectations do targeted individuals and organizations have regarding exporting and importing as a means of bridging their own assemblies of various tools and systems?

Which collections of interaction objects in your application concepts might targeted users want to export? What data might they like to import?

What data formats could support directed, manual exchanges of content between computing tools? What open standards could allow workers to move data into forthcoming and future applications — some of which your team may not yet know about or be able to predict during your own product development process?

What flexibilities could allow workers to have desirable levels of control over the content that is moved into and out of your application concepts?

What vulnerabilities and security problems might be created by allowing the import of "outside" data? What functionality responses could mitigate these risks?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K10. Openness to Application Integration and Extension

In order to better support their local processes, knowledge workers and their organizations may want to effectively combine different applications or add to a computing tool's functionalities. Product teams can envision technical features and support that could facilitate integration, or customized functional extension, of their application concepts.

Examples from three knowledge work domains:

> A financial trader no longer has to cut and paste information between his trading application and a secondary tool. His firm's IT staff has coded an automation that has removed these frequent, tedious tasks from his day, opening up more time for him to focus on trading and analyzing market trends **(see illustration on next page)**.
>
> An architect asked her IT department to develop a small addition for their studio's building modeling application. This new option allows her to view building model data in the context of project budgeting data.
>
> A scientist makes vendor selection decisions based in part on her desire to have her entire clinical lab's operations integrated into a unified clinical data repository. She looks at potential products as collections of functionality that could be integrated into that central system.

Given that a single application is often only one component of an overall computing system (A5), technically integrating various tools together can become a desirable scenario. Organizations can also build unique functional extensions within an application's framework to meet local challenges inside the context of their valued tools (A8, K13). These needs based, technical interventions can positively alter the essential character of knowledge workers' experiences. For example, integrating two products together can offload manual interoperation efforts (E3, E4, K9) that would otherwise detract from workers' less tedious, higher value efforts (D3, D4).

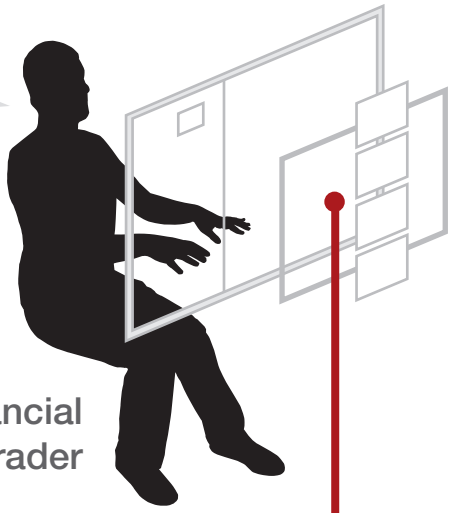
Product teams can facilitate these integrations and extensions by envisioning concepts for "opening up" certain facets of their applications' inner workings to targeted audiences. For example, teams can plan to publish documented (K7) technical interfaces and code for well defined points of flexibility, which could then be used in customer organizations, or larger communities of practice, to make desired systemic connections and custom improvements.

When product teams do not actively consider how individuals and their organizations may want to combine and extend new tools, resulting applications may be experienced as inaccessible, unchanging "islands" of functionality (M4). From knowledge workers' perspectives, these "closed" offerings can become critical barriers to long term productivity and satisfaction (D2, D3, G3).

Conversely, from a product firm's perspective, extensive integration can lead to a situation where the brand of their offering becomes "too invisible" to individual users (C1, C2, L4). In a similar vein, a product's top down strategy may not be conducive to many types of integration or extension scenarios, despite their potential value for targeted individuals and organizations. Definers and designers can attempt to balance "protective" thinking with a desire to support open tailoring and innovative evolution of their product within their user community.

See also: A, B, C8, E, I5, I6, K, M

A certain amount of trading is just like George Jetson work, doing routine things over and over again...



And our group is always trying to figure out how to remove that work so that we can spend more time making deals...

FULLY INTEGRATED APPLICATION AREAS

For example, we had our IT group integrate some key parts of our two main tools, even though they are made by different companies.

That integration saves us traders a lot of copy and paste work — and that's exactly the kind of work that computers should do for us, not the other way around, right?

Key *application envisioning* questions:

Which of the work practices that your team is striving to mediate could bridge multiple computing tools in knowledge workers' technology environments? Where might custom functional extensions address unsupported needs? What specific, publicized points of technical openness could allow target organizations to locally recombine and add on to your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which computing tools do targeted individuals primarily use, given the larger constellation of technologies that are available to them?

What role do each of these tools play within your team's targeted tasks and larger activities?

How do knowledge workers currently coordinate their various onscreen applications to accomplish their goals in different scenarios?

Which interoperations between computing tools are frequent and standard enough — and outside the core of enjoyable, valued, thinking work — to represent opportunities for integration and automated interoperation?

What integrations and custom extensions are already present in targeted workers' environments? Who accomplished these highly technical feats? What successes and problems did they have along the way?

Where might your team's sketched strategic directions suggest "open" and networked approaches to other technologies in targeted workers' environments?

At what point might your product and brand become "too invisible" due to integration or "too mutated" by local modifications?

What market trends and technological realities might your team consider when ideating around the technical openness of your application concepts?

What attitudes and expectations do targeted individuals and organizations have about how open their chosen tools should be to systemic connection and custom improvements?

What features might allow your application to become a platform for innovative local solutions? What specific points of technical openness could provide value?

How might your team effectively document and publish technical interfaces and code? What additional support and services might you provide?

What vulnerabilities and security problems might be created by opening up your product to outside integration and extension? How could your envisioned design responses mitigate some or all of these risks?

How might your team eventually learn from the changes that your users will presumably make based on technical openings in your computing tool? How could you promote community sharing of these local innovations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K11. End User Programming

Product teams can envision functionality concepts that could allow knowledge workers to program different sorts of coded routines within their computing tools, such as the steps followed by an automated process. Interactive, task specific methods can make programming straightforward in the context of workers' own goals and technical skills.

Examples from three knowledge work domains:

> A scientist writes her own algorithm within her analysis application to visualize the data from one of her lab's clinical research studies. She starts with a standard algorithm provided by the tool, then exploits defined flexibilities to modify its rules toward the new visualization approach that she wants to try **(see illustration on next page)**.
>
> A financial trader is reviewing performance trends in his group in order to find potential areas for improvement. To locate only deals made within the last month that match highly specific criteria, he uses his trading application to compose and save a long and complex Boolean query.
>
> An architect selects an option to have her building modeling application record her actions. Once she has finished recording the interaction sequence, she can then easily apply the same sequence of steps to other objects within the same project.

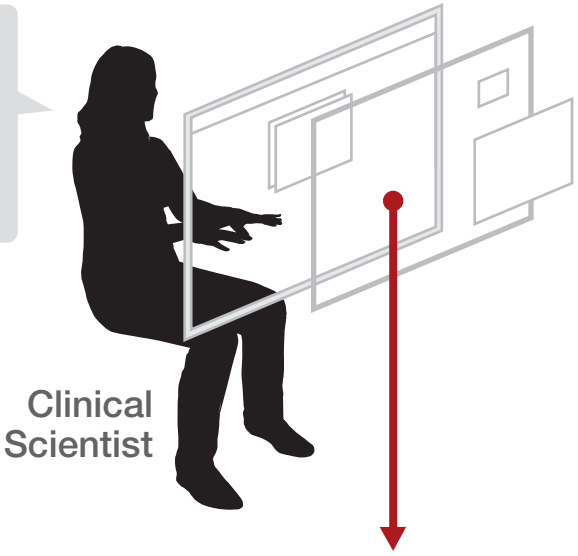
When confronted with exacting and effortful processes (D2, D3), knowledge workers may want to program specialized additions and modifications to their computing tools that are tailored to meet their individual, local needs (E, A7, A8). They may, however, find the notion of tackling even small programming efforts to be daunting. For example, while functionality for coding macros can allow workers to extend some contemporary products in diverse ways, it may require them to invest significant effort in order to learn unfamiliar and abstract programming languages.

Since many people are not accustomed to writing code like a computer scientist, product teams can envision tailored programming methods that encapsulate known, inherent rules in targeted work practices and that are intrinsically appropriate for users' technical skill sets. For example, end user programs can be constructed from sequences of restrictive, easily understood "blocks" that are relevant to workers' goals and mental models (I2, I3). Alternately, more implicit methods, such as action recording functionality can allow workers to program their tools based on recorded interactions within a product's "everyday" interface, which can then be "played back" on other interaction objects (B1).

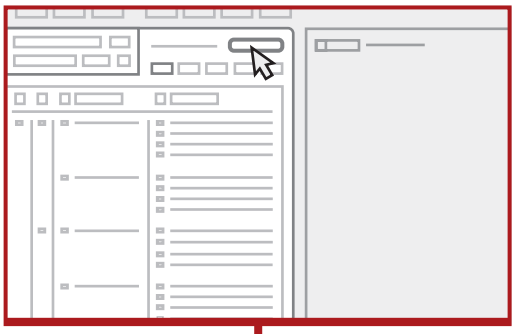
When product teams do not actively consider potential needs for end user programming in their application concepts, resulting products may not be flexible enough to effectively support the diversity of workers' practices. In some instances, completing certain tasks or larger activities may simply not be feasible without customized automation. In the absence of such functionality, product teams may receive a seemingly unsupportable variety of conflicting automation requests, each representing a granular need in local, adopted practice (M4).

See also: A, C8, D4, I1, K, M

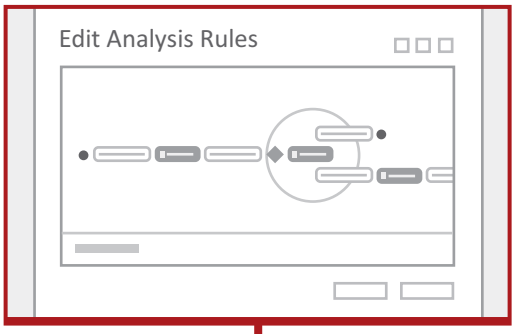
This analysis program has standard routines to transform data, but there's always some other transformation that I want to do...



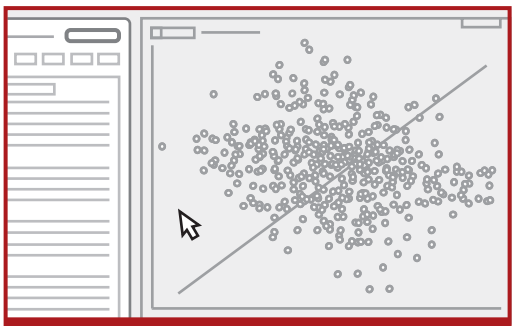
So I'm going to start with one of the rule sets that the product came with, and I will see if I can't change it to analyze how I want it to...



The existing rules are written in a sort of plain language of symbols and text that I can change or add on to...



And now I'm using my new routine with this scatterplot, and the data looks very different. This could be very good...



Key *application envisioning* questions:

What functionality concepts might your team envision to allow targeted knowledge workers to create their own algorithmic rules in order to meet local and emergent needs? What inherent constraints, representations, and interaction idioms might you draw upon to promote clearly bounded and intuitive "coding" experiences?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What do targeted individuals and organizations think of any end user programming features in their current computing tools?

What programmed extensions have they created using these features, and who created them?

Which of the work practices that your team is striving to mediate contains important variabilities in rule based processes?

What types of changes might targeted knowledge workers want make to the default algorithms in your sketched application concepts? Are these changes predictable enough to become customization options — or are they diverse yet important enough to warrant some kind of programming flexibility?

Might programming needs be so infrequent and unchanging as to make technical openness to extension by IT staff more desirable than end user programming options?

What larger technology trends and advanced analogies to other products could valuably inform your team's envisioning of potential programming concepts?

What interactive programming methods could allow users to flexibly "code" algorithms in ways that are appropriate for their skills and the targeted domain context?

How might your sketched functionality ideas for end user programming allow individuals to carve out particular yet easily understandable paths through well characterized spaces of rule based possibility?

How might people test out their own algorithms without any risks to valued data? How could these testing outputs reference your product's larger error prevention and handling conventions?

How might your team eventually learn from the changes that your users will presumably make via programming options in your computing tool? How could your product's overall system promote community sharing of these local innovations?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K12. Trusted and Credible Processes and Content

When knowledge workers are confident that an interactive application follows known professional standards or was contributed to by credible sources, they may be more likely to trust the computing tool's processes and content. Product teams can envision honest and direct ways to engender these cues in their application concepts.

Examples from three knowledge work domains:

> A financial trader knows that highly respected people at his firm provided feedback that was incorporated into the new version of his trading application. This makes him more confident that the tool's interactive approaches and underlying rules are well suited to his work practices **(see illustration on next page)**.

> A scientist views the algorithms supplied by her analysis application as credible because leading luminaries in her clinical research field, including her own mentors, were consulted during the creation of the product.

> An architect trusts the default reference information about construction materials that her building modeling application supplies because it comes from one of her preferred information sources.

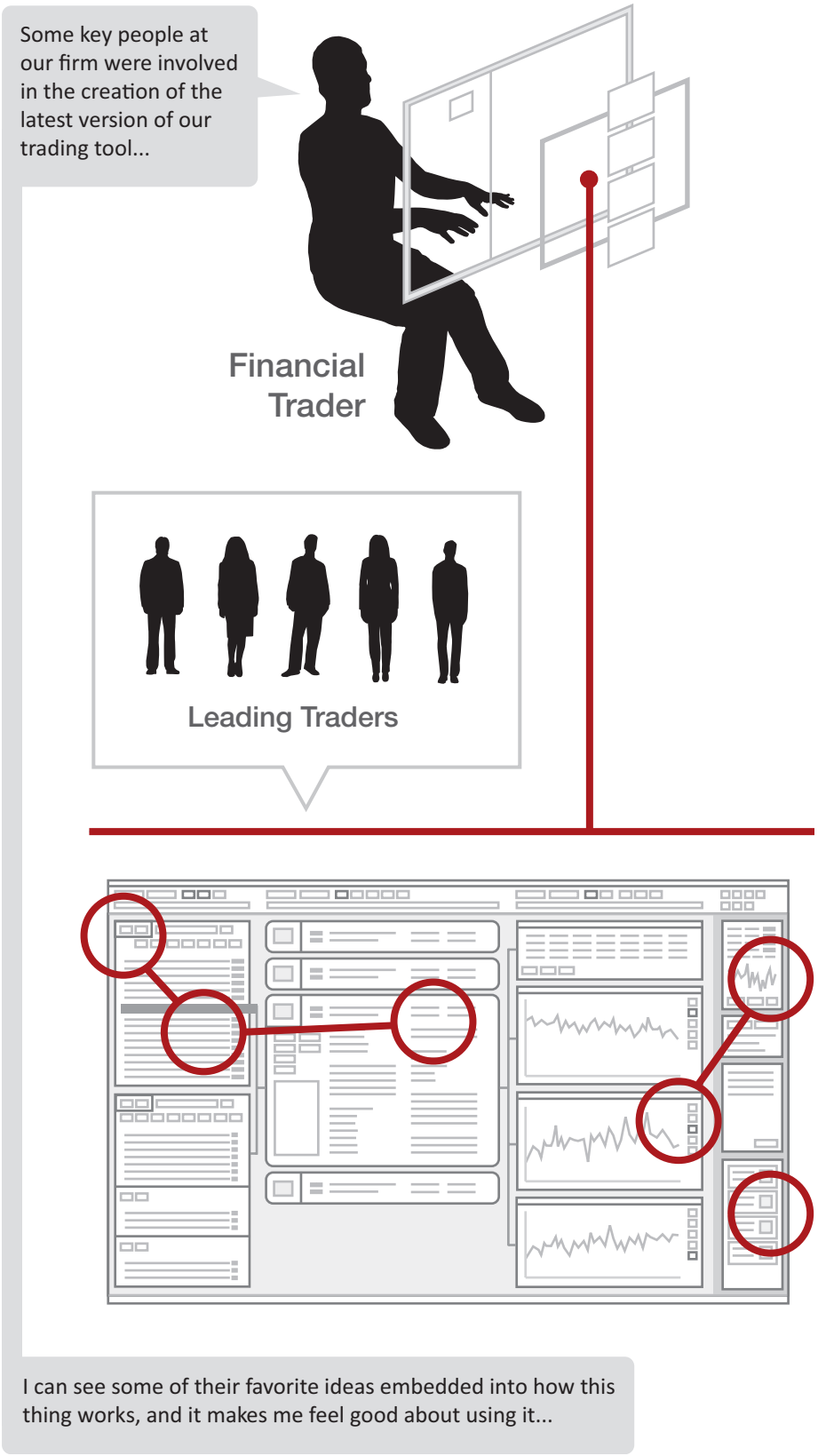
Knowledge workers are often valued for their ability to assess the suitability of potential tools for their own, and their organizations', processes. These skills can be especially important when the introduction of a new computing tool could have significant impacts on the character and outcomes of central work activities (A).

While the high level brand promise of a product firm, product family, or individual application can be a persuasive part of establishing a connection with knowledge workers, the particulars of how an application functions may also be a critical ingredient in peoples' perceptions. All products implicitly communicate some of the underlying assumptions of their creators, and people may question the contributing sources and rationale behind specific definition, design, and implementation decisions that are particularly relevant to their own day to day efforts.

Product teams can promote trustworthiness and credibility by visibly communicating that an application follows accepted approaches and standards within a knowledge work domain. Products can also explicitly cite certain trusted individuals and organizations in targeted professional fields. The involvement of these parties in *application envisioning* can generate valuable insights, and resulting computing tools can meaningfully invoke their contributors' industry cache via the structural embodiment of their leading ideas.

When product teams do not actively consider how they could establish the trustworthiness and credibility of their application's processes and content, knowledge workers may greet resulting products with a healthy dose of skepticism. A lack of credibility can impact both individual and collective attitudes about a brand, which may then strongly influence product acquisition and adoption decisions.

See also: C1, C8, G1, E3, E4, E6, E5, G6, I5, K, M


Some key people at our firm were involved in the creation of the latest version of our trading tool...
Financial Trader
Leading Traders
I can see some of their favorite ideas embedded into how this thing works, and it makes me feel good about using it...

Key *application envisioning* questions:

Which domain standards and thought leaders are viewed as credible by targeted knowledge workers and their organizations? How might your team meaningfully involve certain trusted sources in your ideation efforts, incorporating their input and insights in order to enhance the usefulness, usability, and desirability of your offerings?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What trusted professional standards could be relevant for the work practices that your team is striving to mediate?

What credible individuals and organizations could be valuable contributors and collaborators?

Which of these sources might be a good match for your team's sketched strategic directions?

What risks might occur from aligning your computing tool with particular industry standards and well known individuals? What controversies might be better avoided?

How might your team incorporate pertinent insights and information from certain sources into your sketched application concepts?

What source cues could be visibly incorporated into the structure, content, and interactive processes of your product? What types of source related messaging might be effective within targeted markets?

What positive impacts could these honest and applicable citations have on your offerings' brand positioning?

How might your team involve trusted contributors in your computing tool's eventual user community?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

K13. Reliable and Direct Activity Infrastructure

Interactive applications that perform reliably and give knowledge workers a sense of uninterrupted, direct action have the potential to become "at hand" infrastructure in work activities. To prevent situations where individuals and organizations limit their adoption of unreliable computing tools — or jettison them entirely — product teams can envision early requirements for experienced performance.

Examples from three knowledge work domains:

> An architect has used her new building modeling application for several months, and many of her typical actions in the product now feel like second nature. Since the application has been dependable, she feels comfortable estimating how long it will take her to accomplish specific efforts using the tool **(see illustration on next page)**.

> A financial trader is so used to his trading application working smoothly and without interruption that he becomes furious during those rare occasions when the tool unexpectedly displays a glitch. He is known for having thrown his phone through his screen in a fit of rage after a small but damaging technical problem.

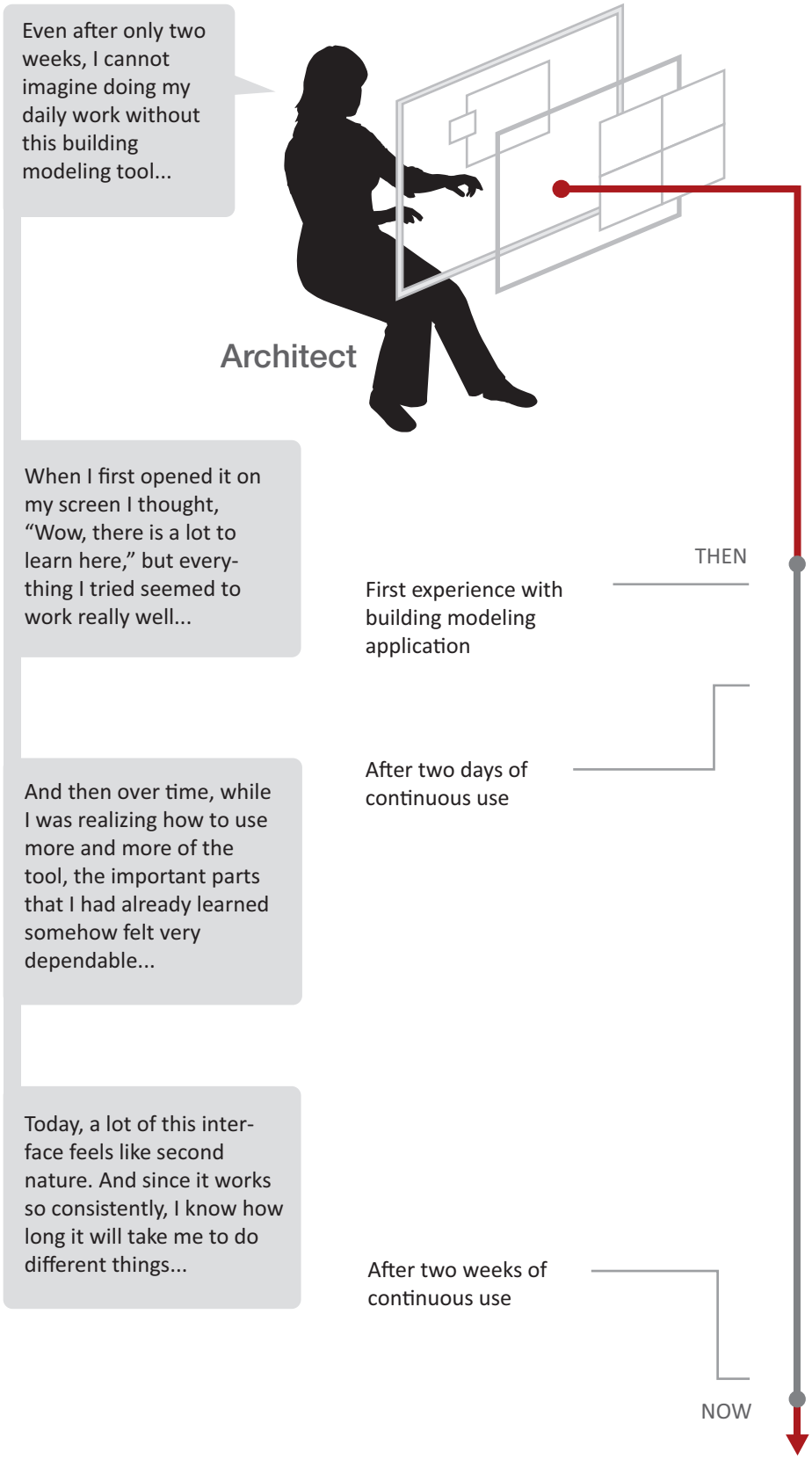
> A scientist likes that her new analysis application does not "freeze up" during large, computationally intensive analysis processes like other, similarly focused products that she has used in the past.

Stable and useful tools can become relied upon infrastructure in knowledge work. After an onscreen application has fulfilled its initial promises and users have begun to develop their own meaning and skills within its framework, sequences of call and response interaction can become highly rehearsed, routine, and even seemingly automatic (D7). Ideally, after workers have adopted an application into their practices, they can simply turn to the "at hand" product to accomplish their goals in a familiar and direct manner (D4, G1).

Product teams can envision high standards for reliability. They can also strive to cultivate a sense of on demand availability and tightly coupled action and reaction within their technologies. At a minimum, teams can identify critical performance areas within their application concepts and set appropriate goals for them. Going further, they can set performance goals for each of the central functionality concepts that they have sketched, benchmarking a desirable sense of flow for their tool's core pathways.

When product teams do not actively consider how their applications could become reliable and direct fixtures in knowledge workers' practices, resulting products may promote frustrating user experiences, even after extended usage. Applications that do not behave dependably may not engender trusted, first choice status, given the availability of other brands or alternate avenues of action (K12, E6). When confronted with inconsistent system behaviors, people may spend additional effort performing "defensive" procedures, such as versioning work (D2, D3, H1). Our human tendency toward sense making around unusual occurrences may cause users to "incorrectly" redefine their conceptual models of how a tool functions (C1). In the worst cases, these alternate conceptions can become "ghost stories" of what to avoid and why it is dangerous, potentially leading to rolling failures in adoption.

See also: A, C, D1, G, H, K, M


Even after only two weeks, I cannot imagine doing my daily work without this building modeling tool...
Architect
When I first opened it on my screen I thought, "Wow, there is a lot to learn here," but everything I tried seemed to work really well...
And then over time, while I was realizing how to use more and more of the tool, the important parts that I had already learned somehow felt very dependable...
Today, a lot of this interface feels like second nature. And since it works so consistently, I know how long it will take me to do different things...
THEN
First experience with building modeling application
After two days of continuous use
After two weeks of continuous use
NOW

Key *application envisioning* questions:

How might the experienced reliability of your team's computing tool instill a sense of confidence in targeted individuals and organizations that could lead them to adopt its options into the structure of their work? How might your functionality concepts provide a sense of direct, low latency action on the objects of workers' goals?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What requirements do targeted individuals and organizations have for the stability of their computing environments? How far do they currently go to prevent reliability issues in their IT infrastructures?

How well do their current onscreen applications live up to these standards?

What can be learned from understanding reliability problems in similar products? What has prevented other computing tools from being successful in markets similar to the niches that your team is targeting?

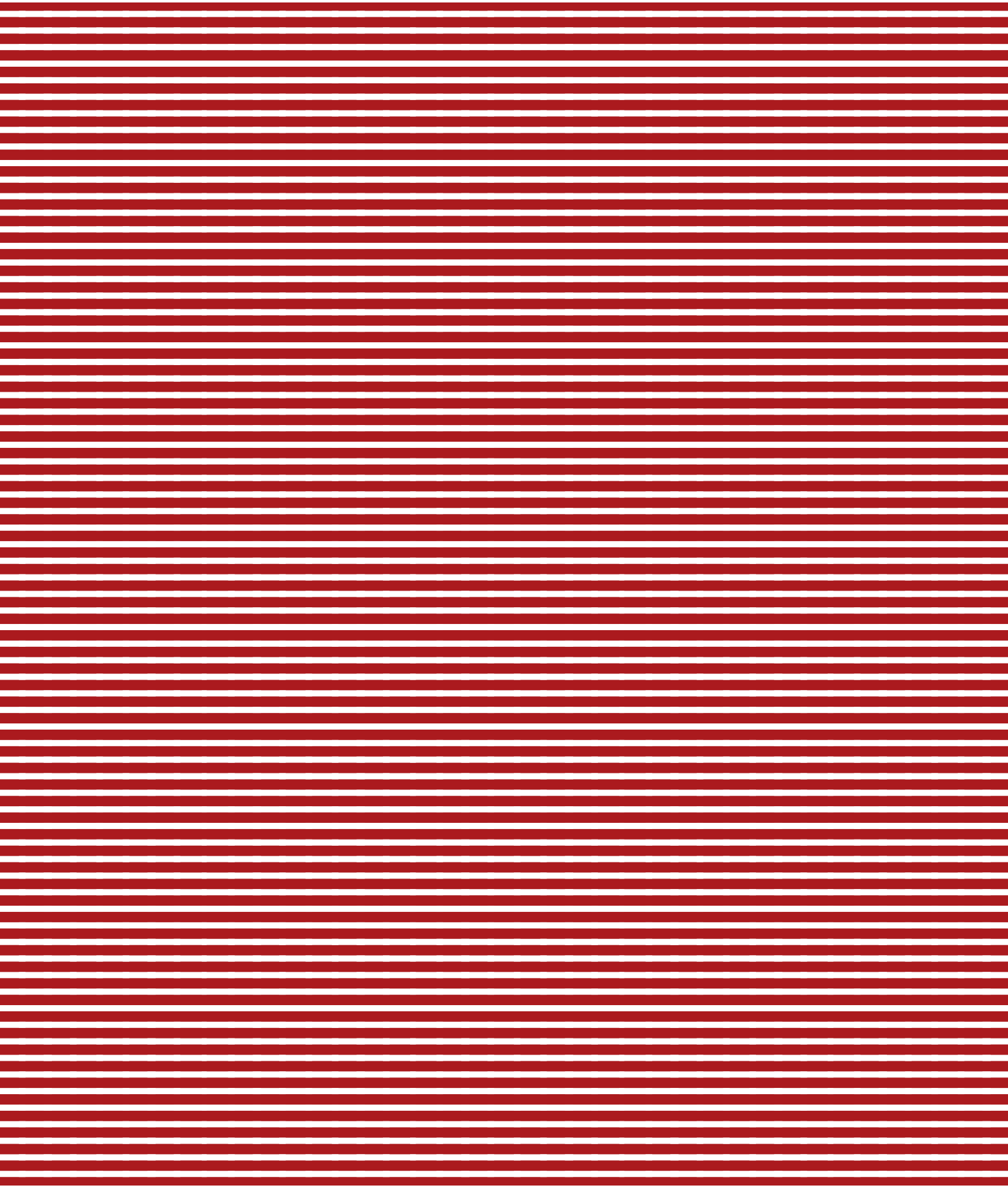
What advanced analogies to high quality products might your team draw upon when thinking about the reliability of your offerings? How might these "gold standard" stories influence your ideas about baseline performance for your computing tool?

Which primary functional areas in your sketched application concepts are critical to envision as high performance interactions? How might these areas promote compelling experiences of direct availability and tightly coupled action?

Which of your sketched functionality concepts might not have such high performance requirements, based on frequency of use or other factors?

What risks might your product face if it does not eventually meet your team's envisioned levels of performance in implemented reality?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?



L.

Pursuing Aesthetic Refinement

Valued computing tools can desirably communicate with knowledge workers on an emotional level, delighting users and creating a sensory environment that is conducive to focused thinking.

Designing such compelling aesthetics requires the critical examination of a product's formal qualities, behaviors, and larger positioning.

During *application envisioning*, product teams can map and explore potential aesthetic meanings and refined aesthetic directions.

By taking time to locate and generate relevant emotive qualities for their onscreen tools, teams can uncover opportunities for more appealing, recognizable, comprehensible, and brand differentiated experiences.

Generally speaking, product aesthetics have traditionally been an afterthought in the development of computing tools for the workplace. This positioning often stems from the fragmented specializations within product teams, a demonstration of the implicit priorities of project groups staffed with a proportionately high number of technically oriented engineers. Perhaps more importantly, the case for emphasizing aesthetics can be difficult to make in many highly specialized and niche markets, where the bar for product design is often set very low.

At the time of writing, explorations of aesthetic meaning in complex and technical interactive products — beyond surface level branding — have received relatively little attention. Although aesthetic decisions contribute to brand, not all aesthetic ideas need to be driven exclusively from top down ideas about literal brand attributes. Application concepts can meaningfully reference aesthetic ideas from targeted knowledge work domains in order to invoke workers' existing understandings and better situate products within work contexts.

Going a step further, product teams can envision the aesthetics of their application concepts with pleasure, engagement, and workers' sense of accomplishment as principle motivations. While these goals may not weigh as heavily as they would, for example, in an entertainment technology, there is nothing inherent to knowledge work products that requires them to be so far behind consumer products in meaningful and desirable aesthetic refinement. Teams can strategically reconsider established aesthetic priorities within their targeted markets, especially in mature product genres where new sources of value and differentiation are at a premium.

This category contains 5 of the 100 *application envisioning* ideas in this book:

L1. High quality and appealing work products

L2. Contemporary application aesthetics

L3. Iconic design resemblances within applications

L4. Appropriate use of imagery and direct branding

L5. Iconoclastic product design

Product teams can use these ideas to explore overriding aesthetic approaches for their application concepts, which can subsequently be used to inform aesthetic decisions throughout detailed design and implementation. Early ideation focused on those overriding choices, rather than post hoc efforts during the mid or final stages of a product's development, can help teams uncover innovative opportunities and promote a unifying clarity in application design.

The central notion of this category is most closely related to the "Defining interaction objects" (B), "Establishing an application framework" (C), and "Enhancing information representations" (F) categories.

L1. High Quality and Appealing Work Products

Knowledge work typically results in artifactual outputs that are communicated to others, which recipients may then use to understand work progress and evaluate its outcomes. Product teams can envision functionality concepts that could make it easier for users to generate desirable work products with refined aesthetics.

Examples from three knowledge work domains:

> A financial trader likes that when he sends a specific offer or completed trade form as an email attachment, his trading application automatically creates the document in the professional "look" of his company **(see illustration on next page)**.
>
> An architect likes that she can tailor the visual conventions that her new building modeling application uses when it automatically creates design drawings, specifications, and other documentation. This flexibility allows her to maintain a strong continuity with how her studio has traditionally presented their work to long standing clients and other audiences.
>
> A scientist is surprised by the quality of the graphs that she can export from her analysis application. Previously, to get high quality visuals that appropriately represent her lab's work, she had imported different subsets of clinical data into separate, specialized graphing tools.

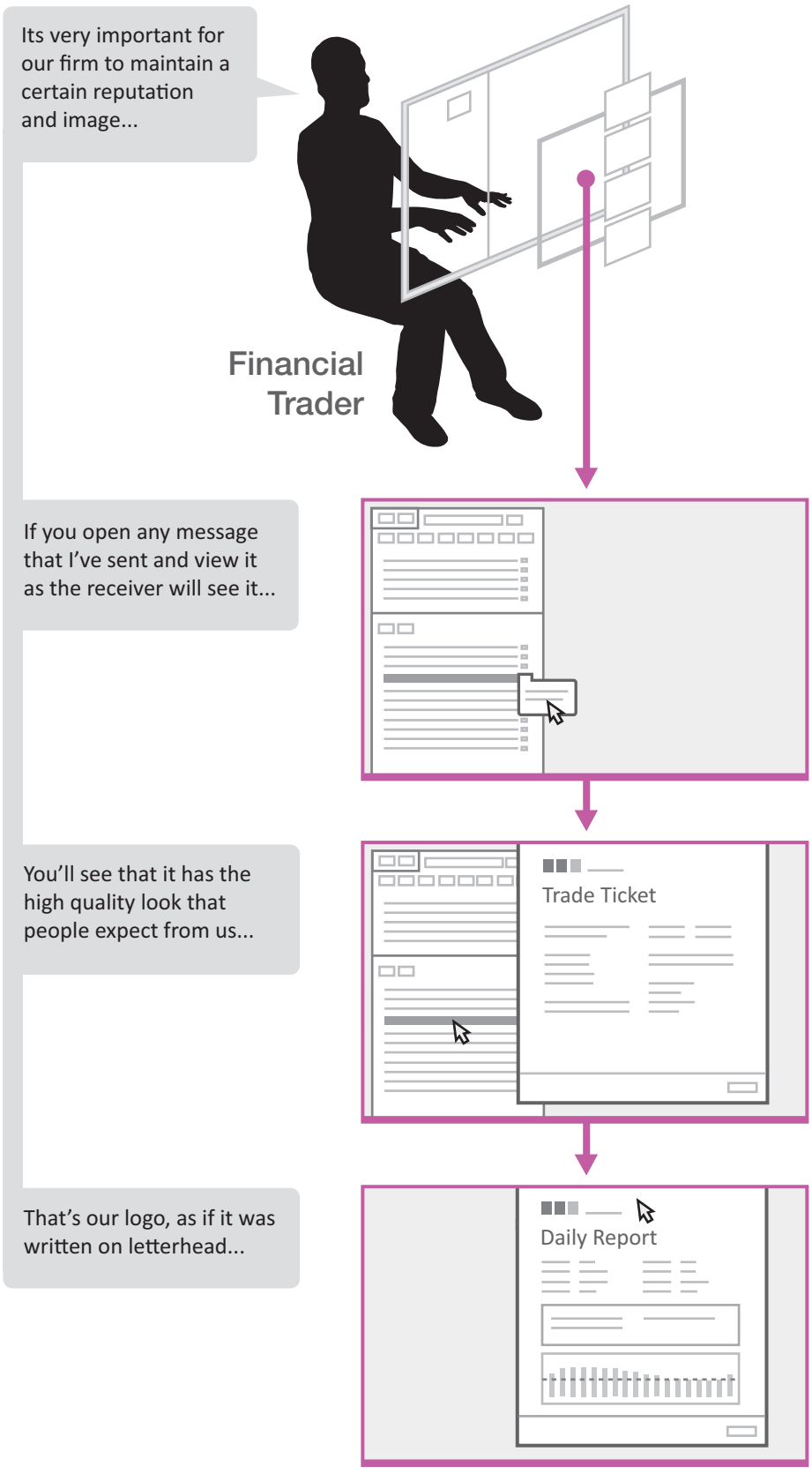
In some knowledge work domains, work products may be the culmination of countless hours of effort. In other domains, they may be created relatively rapidly and repeatedly, in highly standardized formats. In either case, the stakeholders of an activity may only see workers' artifactual results (G7, J). Since many individual efforts and stages in knowledge work (A) can become largely invisible once they are completed, workers often place a high priority on the content, format, and appearance of their outputs.

In order to envision functionality concepts that could reduce the effort needed to create valued work outputs (E3, E4), product teams must understand related professional practices, standards (A4, F2, J6, K3), and flexibility requirements (A9). Appropriate aesthetic directions can emerge from meaningfully referencing information artifacts that targeted workers currently use while at the same time enhancing their graphic design clarity (F, L3).

When product teams do not actively consider support for creating high quality and appealing work products in their application concepts, they may overlook opportunities to provide value at the iterative and concluding seams of workers' activities (G1). Although users may prize a resulting tool's functional offerings for getting their work done, they may then need to expend considerable additional effort to create outputs that they consider to be desirable (D2, D3). Since many knowledge workers do not necessarily have nuanced design skills, their own creations may not effectively communicate their important content.

Conversely, not every application needs to include extra functionality for creating appealing work products. When constraints vary widely (A6, A7, A8, K1), documentation efforts can be offloaded through seamless interactivity (K8), interoperability (K9), or integration (K10) with workers' related computing tools.

See also: B, C5, C8, E, K12, L, M


Its very important for our firm to maintain a certain reputation and image...
Financial Trader
If you open any message that I've sent and view it as the receiver will see it...
You'll see that it has the high quality look that people expect from us...
Trade Ticket
That's our logo, as if it was written on letterhead...
Daily Report

Key *application envisioning* questions:

What types of artifacts are created in the knowledge work practices that your team is striving to mediate? How might your computing tool offload some of the effort of generating certain outputs while at the same time enhancing the effectiveness and appeal of their design?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Which work products do targeted individuals currently put the most emphasis on? The least emphasis?

What roles do specific artifactual outputs, whether physical or digital, play in targeted work practices?

What audiences are different work products intended for?

Is there a separation between informal, working versions and formal, presentation versions of these artifacts? How do professional practices and standards commonly vary for different levels of formality?

How are different types of outputs currently delivered?

What do targeted knowledge workers think about their current processes for creating and communicating their own outputs? What parts of these processes require tedious effort or can frequently lead to breakdowns? Could these problems represent potential opportunities for your product?

What do recipients of work products think about current artifacts?

What work products might become the natural "take aways" of your team's application concepts? What new types of outputs could provide value as part of your strategic approach to mediating work?

What functionality concepts might your team envision to automate standard operations in the process of creating certain work products?

What media formats could be appropriate for each type of work output that your team is envisioning? What technology implications might these choices have, in the context of current trends and existing systems?

Which of your sketched interaction objects could certain outputs be based upon? How might work products retain recognizable representational features from these sources?

How might your team valuably enhance the graphic design clarity of important work products, without any extra work on the part of their creators?

What flexibilities might targeted individuals and organizations want or need in order to make the automatic generation of work products useful and relevant in their own, local cultures of practice?

What contextual pathways to communication options could conclude the process of creating work outputs in your sketched application concepts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

L2. Contemporary Application Aesthetics

The stylistic aspects of conventional onscreen interaction and visual design have changed over time and will continue to do so. Product teams can promote learnability, as well as attributions of product quality and utility, by envisioning usages of contemporary user interface aesthetics in their application concepts.

Examples from three knowledge work domains:

> An architect recognizes many of the user interface conventions of current architectural software in her new building modeling application. Although the new tool is substantially different, it has elements that somehow look similar to the latest versions of other tools she has used, and she feels that these stylistic cues help her to "know where to go" **(see illustration on next page)**.

> A financial trader recalls how his firm's trading application has gone through several different user interfaces over the years, with each iteration reflecting visual "best practices" of the times. He is somewhat of a gadget connoisseur, and he likes using tools that feel up to date.

> A scientist is used to working with applications that do not look very current, even when they are newly released. Her new analysis application is an exception, and she is surprised by how much she enjoys its emphasis on modern "look and feel."

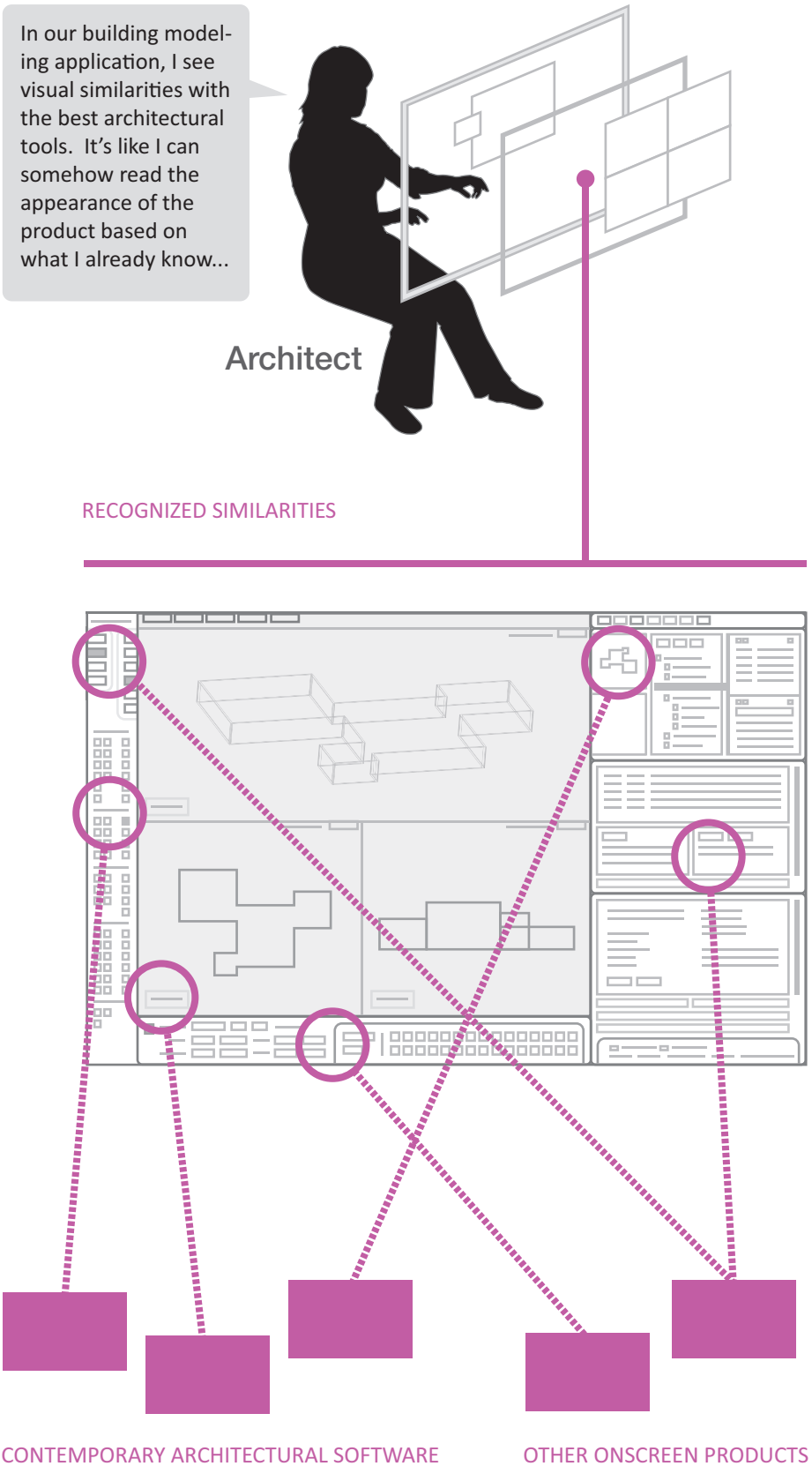
Contemporary design standards for computing tools are, at least in some part, a moving target. While certain foundational interactivity conventions of graphical user interfaces appear to be here to stay, a variety of stylistic variations on those conventions are continually emerging in new products, both within interactive computing at large and within a breadth specialized domains (C2, C3, D7, F2). Applications are also increasingly subject to purely graphic trends, and users' judgments of a product's visual design currency may drift to reflect contemporary styles (L).

While enduring tools for work should probably not become purely fashion artifacts, not all interface standards remain contemporary, and dated conventions can disrupt a product's design vocabulary. By explicitly mapping current and emerging application design trends, product teams can select contemporary approaches and standards that suit the activities that they are striving to mediate and generally match targeted workers' preferences (A).

When product teams do not actively consider how they might reuse contemporary user interface standards, opportunities to appropriate established interaction styling and visual languages (F10) in a consistent manner can be lost. Workers may perceive resulting tools as being less relevant or industry leading (K12), which — depending on a product's competitive environment and its other sources of proposed value — may lead to negative overall halo effects.

Conversely, envisioning interfaces from a narrow perspective of current, "permissible" standards can exclude more learnable (K2, K6) and otherwise optimal (D2, D3, D4) interface conventions. It may also detract from lines of design thinking that can result in more compelling, iconoclastic designs (L5).

See also: A, A4, C8, F, J6, K1, M


In our building modeling application, I see visual similarities with the best architectural tools. It's like I can somehow read the appearance of the product based on what I already know...
Architect
RECOGNIZED SIMILARITIES
CONTEMPORARY ARCHITECTURAL SOFTWARE
OTHER ONSCREEN PRODUCTS

Key *application envisioning* questions:

What current and emerging trends in user interface aesthetics could be relevant for your team's targeted markets and the work practices that you are striving to mediate? How might your team distill selected contemporary interaction and visual design directions into stylistic conventions that could be applied across your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What do targeted individuals think about the aesthetics of the onscreen tools that they currently use?

What contemporary design standards are they familiar with in the context of their day to day work? In their use of personal technology, outside of their working lives?

What larger stylistic trends in application design might your team map and make predictions around?

What types of emotional connections between user and brand are central to your design strategy and application concepts?

Which contemporary aesthetic approaches in interaction and visual design could promote these types of connections and attributions?

Which contemporary approaches might targeted knowledge workers perceive as being appropriate and appealing for their own day to day visual environments? How might they identify with certain aesthetic directions in the context of their working lives?

Which approaches might be too unproven, inefficient, or edgy for your application concepts, given that your team is striving to create a highly functional tool for thinking work?

How might certain interactions with relatively "known" onscreen aesthetics promote emotional responses that are conducive to attentive, focused thinking?

How might your team's choices about contemporary interaction and visual design aesthetics play out across your sketched application frameworks, information representations, and functionality concepts?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

L3. Iconic Design Resemblances within Applications

Knowledge work domains have visual cultures of iconic designs and related products that have evolved over time. Product teams can leverage those familiar cultural understandings to give their onscreen elements intangible, or outright meaningful, family resemblances with known artifacts.

Examples from three knowledge work domains:

> A scientist feels that the look of her new analysis application "fits into" her clinical research work. It has a white, color coded, clearly labeled feel that reminds her of the simple and informative labels on her lab's reagent containers and packages **(see illustration on next page)**.
>
> An architect sees a visual resemblance between the elements in her building modeling application and offline tools found on a drafting table. For example, the application's semi-transparent menus are the same milky color as the old, semi-transparent protractors in her desk drawer.
>
> A financial trader has noticed that a number of key trading applications are starting to look more and more like a startup firm's market information application, which has had an industry leading influence on design. The style somehow looks very "finance" to him, in a good way.

Knowledge work professions can have rich visual histories of iconic artifacts that have become emotionally symbolic of their work culture (A1, K1, F, J6). Given that workers often interpret new tools from the vantage points of known artifacts and existing approaches (C1), targeted references to earlier visual forms can summon useful, pleasing, and comfortable associations.

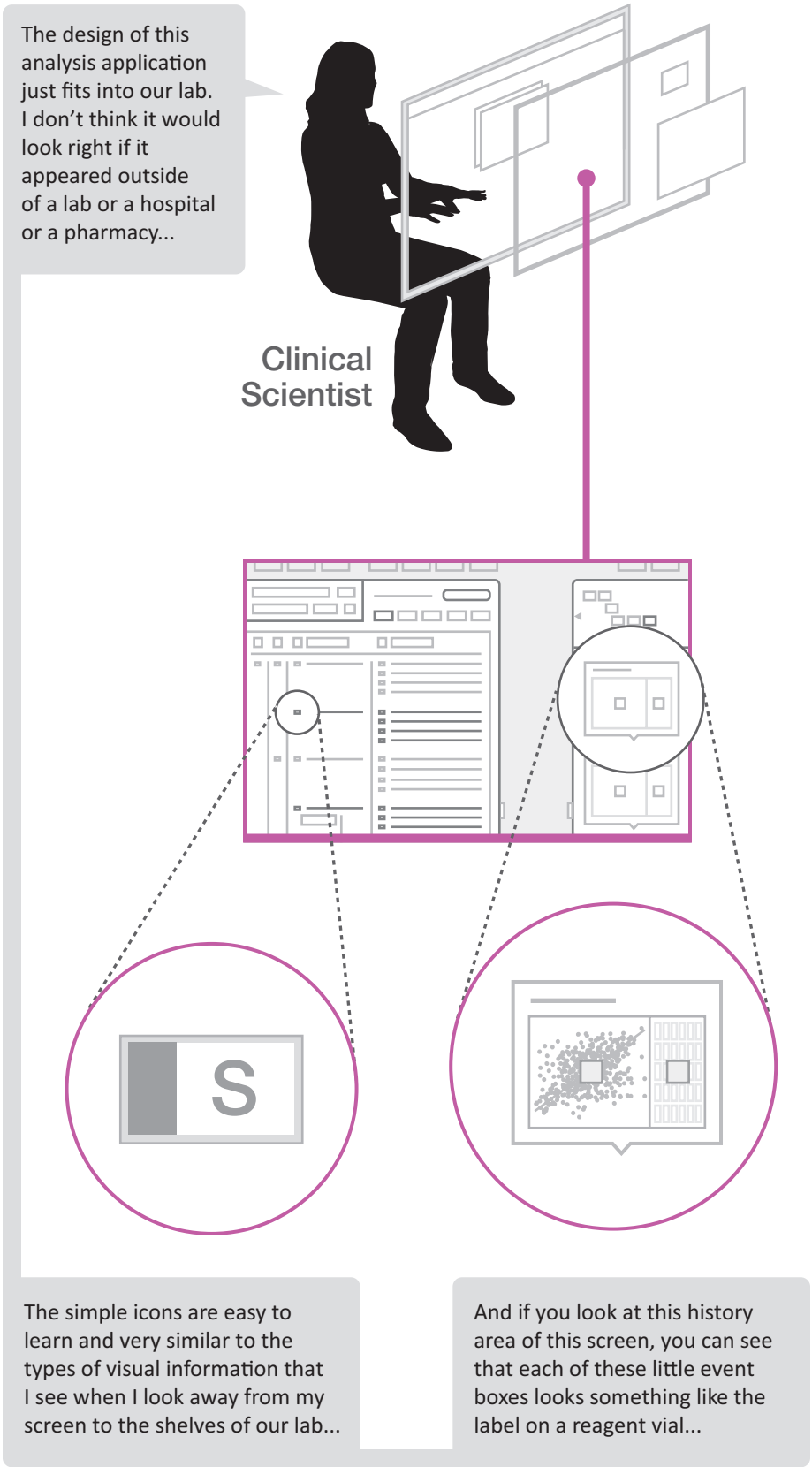
Product teams can intentionally situate their interactive applications within targeted cultural contexts by envisioning diverse design approaches that directly display, or indirectly evoke, affinities with established domain artifacts.

Although teams may find it difficult to incorporate iconic resemblances at the level of an entire application (C2), clearer opportunities may present themselves at the level of individual functions (C3) or interaction objects (B1). These references can be envisioned within an application's top down brand approach to ensure that individual cues are part of a larger, cohesive, aesthetic (L).

When product teams do not actively consider how iconic design resemblances could be incorporated within their application concepts, opportunities to make intentional use of workers' existing artifactual literacies can be lost. Without these cues, resulting applications may feel more generic to their targeted audiences. Individual users may find it more difficult to apply their specific cultural understandings to such products (D2, K2, K6), making it more difficult for them to recognize that these tools are intended for their own work practices (K3).

Conversely, some iconic references do not translate well into a contemporary interactive application (A3, F2). Workers may associate certain artifactual cues, such as dominant and literal metaphors, with ways of working that are no longer relevant to them or generally not conducive to being enacted on a computer screen.

See also: A, B3, K5, M


The design of this analysis application just fits into our lab. I don't think it would look right if it appeared outside of a lab or a hospital or a pharmacy...
Clinical Scientist
S
The simple icons are easy to learn and very similar to the types of visual information that I see when I look away from my screen to the shelves of our lab...
And if you look at this history area of this screen, you can see that each of these little event boxes looks something like the label on a reagent vial...

Key *application envisioning* questions:

What iconic artifacts are part of the visual and material culture of targeted knowledge workers' day to day professional environments? How might your sketched functionality concepts and interaction objects subtly or directly reference these artifacts in ways that are both compelling and evocative?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What iconic designs and products do targeted individuals find especially interesting and meaningful? Which workplace artifacts do they not feel an affinity for?

What historical designs in targeted cultures of practice are still iconic, even if they are not currently used?

What do workers attribute to those iconic artifacts that they value? What have these objects come to mean? What emotional connections do they hold?

Which features of certain iconic artifacts stand out to knowledge workers?

How might your team meaningfully incorporate chosen aspects of iconic designs into your application concepts?

Are there any opportunities to draw sweeping parallels between your envisioned product and an existing, iconic product? What impact might those iconic parallels have on your application's strategic direction?

Which references to iconic artifacts could be appropriate for your product's emerging brand? What impact might these references have on brand attributions?

What smaller resemblances to iconic designs could be valuable in your sketched functional areas?

How might iconic references lead to inappropriate interpretations of your product's role and functioning? Where might more generic design approaches be appropriate for what your team is attempting to accomplish?

How might certain iconic cues relate to larger user interface conventions within your sketched application concepts? Which references could potentially provide value as part of your team's reusable interface patterns?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

L4. Appropriate Use of Imagery and Direct Branding

Thoughtfully applied branding and non-interactive imagery are often noticeably absent in computing tools for knowledge work. Product teams can envision how aesthetic treatments and added graphic elements could help build emotional connections with users, promoting brand recognition and appeal while at the same time improving individuals' understandings of product functionality.

Examples from three knowledge work domains:

> An architect enjoys the richly animated imagery that covers the log in screen of her building modeling application. Once she has logged into the tool, however, its interface becomes a quiet, cool, unobtrusive frame that does not distract from her focused attention on the building model that she is currently working on **(see illustration on next page)**.
>
> A scientist likes that some dialogs in her analysis application use complementary illustrations to explain certain ideas, instead of just presenting text. Even after she has learned the illustrated information, she still finds the colorful images to be appealing, and they sometimes act as landmarks that help her find her way.
>
> A financial trader finds the branded color scheme of his current trading application more appealing than the black and green mainframe screens that he used to use. There's something about the new tool's animations and useful, attention grabbing cues that really set it apart from other trading products. It is all very simple, and it does not get in his way.

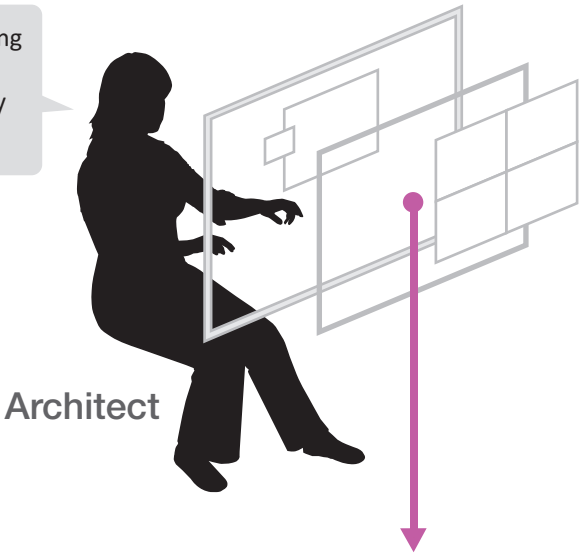
Many contemporary products for knowledge work are aesthetically bland, unrefined, and undifferentiated from one another. While product teams frequently prioritize certain types of iconography and other conventionally "necessary" graphical elements in their applications, other visual treatments and non-interactive imagery may not carry the same weight in teams' design priorities.

Specialized computing tools can have unique, recognizable visual characters. Using the term branding to mean more than just a logo or swatch of color, product teams can extensively brand knowledge work applications while enhancing product usefulness and usability (D4, F). When branding is addressed during early ideation of potential application concepts, teams can envision approaches that broadly embody brand in a holistic sense (A, K12). Definers and designers can also identify opportunities in their functionality concepts for informative, stimulating, and memorable imagery (F10). This content can appear as a part of introductory instruction (K2), infrequently accessed processes (K5, K6), tasks that involve waiting (D3), and other classes of potential user experience.

When product teams do not actively consider how they might incorporate supplementary imagery and direct branding into their application concepts, opportunities to drive useful, unified, and meaningful visual approaches can be lost. While some visual treatments and graphic elements can be added on an item by item basis during product implementation, an early, foundational emphasis may be necessary for teams to arrive at consistent, strategically differentiated, and industry leading design approaches (L5).

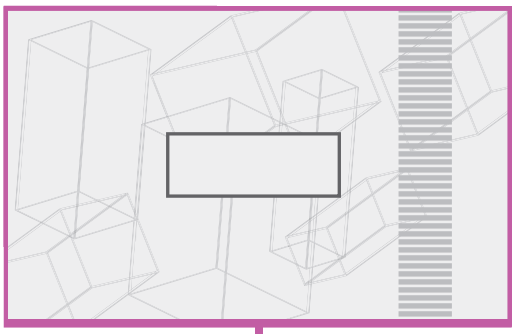
See also: K1, L, M

Starting up my building modeling application has this great, slightly cinematic feel...

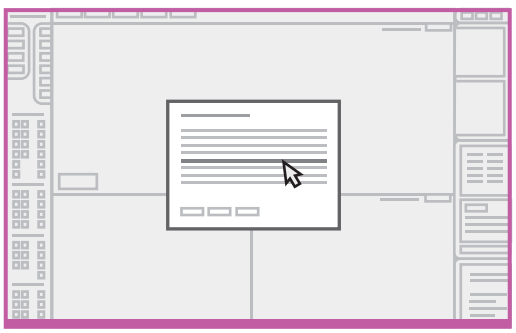


The log in screen comes up fast, and I enter my information...

And while it's loading, there this great animation going on in the back-ground that sort of gives you an indirect feel for what the product does. It feels like all of this slick, integrated building data...



But it's just a quick thing, and it's important that it fades as soon as it can to leave me with the details of my tool and my work...



Key *application envisioning* questions:

How might your team's application concepts be extensively and recognizably branded, while enhancing — not distracting from — onscreen clarity and utility? Where in your sketched functionality ideas could there be opportunities for useful, stimulating, and memorable supplementary graphic elements?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How could your team's interpretation of branding expand to mean more than just a logo or swatch of color?

What types of emotional connections between user and brand are central to your design strategy and application concepts?

What larger design trends could influence your team's approach to of branding and complementary graphic elements in your sketched product?

How might your computing tool have a unique and recognizable visual character, even if that character may emerge from the sum of relatively small design details?

How could selected brand characteristics of your firm, product family, and envisioned offering be usefully and valuably applied to your sketched application frameworks and functional areas?

How might targeted individuals identify with and respond to certain branding approaches, given that your team is striving to create a highly functional tool for thinking work?

What parts of your application's scope have attentional and onscreen constraints that might not be conducive to incorporating any sort of "additional" visual content?

What key ideas, processes, and options within your functionality concepts might be clearly communicated to targeted workers through graphic imagery?

Where might instructive content in your sketched screens benefit from illustrations?

Could the user experiences of starting your product, waiting during specific processes, or exiting the computing tool present opportunities for engaging visual communication?

What styles of illustration could be domain and brand appropriate? How might these styles reference or play against the aesthetic conventions of contemporary computing?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

L5. Iconoclastic Product Design

Many knowledge work applications do not stray very far from the aesthetic mold of "standard" user interface design. Products teams can envision how their application concepts could fully preserve their proposed utility while at the same time gaining uniquely stimulating and emotionally compelling differentiation through novel interaction and visual design approaches.

Examples from three knowledge work domains:

> A financial trader is genuinely surprised by the experience of using his new trading application. He thinks that it is something disruptively different. Its informative, animated transitions and sleek, streamlined, almost "hi fi" appearance makes the trading tool feel something like an advanced, luxurious electronic product — a device that he identifies with and actually wants to use, not just because he "has to" **(see illustration on next page)**.
>
> An architect finds that her new building modeling application looks more like a well designed, humanized medical product than a typical architectural product. While she considers the functional aspects of the computing tool to be more important than its aesthetics, she is surprised by how much she appreciates its refined design qualities day after day.
>
> A scientist finds that her new analysis application transitions through and displays data views in a revolutionary, highly spatial way. She is now less likely to get "lost" while navigating her lab's large clinical data sets.

While the value of iconoclastic product design has been recognized in a range of business sectors, many firms creating computing tools for knowledge work have — at the time of writing — not put a priority on this type of innovation. Some product teams may implicitly believe that disruptively novel design is somehow counter to the goal of creating functional tools for skilled professionals. It is not.

Product teams can sketch and evaluate divergent, iconoclastic approaches for their application concepts. It is important to note, however, that not all aspects of user interface design are ripe for highly novel concepting. Changing some fundamental user experience conventions in the name of revolutionary design can lead to unnecessary confusion and frustration (G1, K5). These fundamentals, which computer users have learned through years of experience, are often rooted in the pioneering design patterns of interactive computing (C3, C9, G2, G3). Outside of these fundamentals, however, product teams can uncover large territories of interactive and visual design convention that are more open to exploration and breakthrough design concepting (C1, C2, F3).

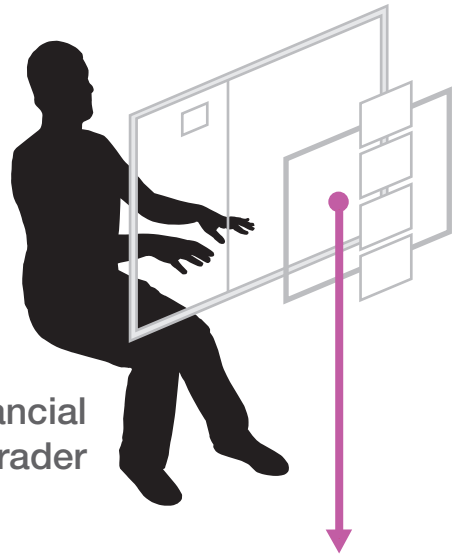
When product teams do not actively consider whether their application concepts could benefit from iconoclastic approaches, opportunities for compelling brand differentiation (L4), beneficial halo effects, and stimulating user experiences can be lost.

Conversely, not all knowledge workers may want to use applications with substantially different design emphases, especially if they perceive novel approaches as potential obstructions to their work outcomes (K2, K3, K6).

See also: A, C, L, F, K1, M

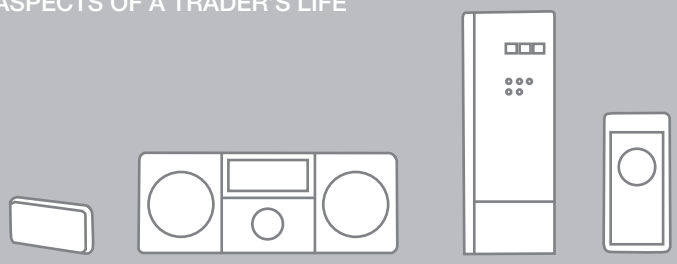
I love the way this new trading tool looks and moves.

It's very different from software as usual. It feels much more designed somehow...



I think that the designers must have thought about the kind of gadgets that I like to own and play with and walk around with...

INNOVATIVELY REFERENCING TECHNOLOGIES FROM OTHER ASPECTS OF A TRADER'S LIFE



I see a lot of small things in this tool that remind me of the types tech that I like to personally buy and use...

It has this quality look that I really want, and I don't feel like I'm sacrificing anything as far as my work goes in order to get this better experience...

Key *application envisioning* questions:

How might your team use your insights into targeted knowledge work practices to sketch truly different, surprisingly engaging, and highly relevant user interface design breakthroughs? What impact could these ideas have on the larger design strategies of your application concepts?

More specific questions for product teams to consider while envisioning applications for knowledge work:

What larger design and technology trends could influence your team's ideation around iconoclastic application proposals?

What types of emotional connections between user and brand are central to your emerging ideas about design strategy?

What might truly iconic differentiation mean for your product? Could it be worth the design investment given your targeted market context?

How might your team usefully focus your desire to do something different, moving beyond simply contradicting the conventional to instead ideate around potentially important opportunities for mediating knowledge work?

How might you use relevant big picture metaphors to think through drastically different design directions?

How might you greatly expand upon small iconic references to artifacts that have cultural significance to your targeted audience? What lines of design thinking could be opened up by certain inspirational reference points?

What advanced analogies to other, potentially unrelated related domains might drive new directions in your team's application concepts?

What novel approaches could clarify specific interactions? How might your team freely re-envision some of your functionality concepts with the goal of promoting more dynamic and engaging user experiences?

What choices about radical design departures make sense given the historical trajectory and brand of your firm, as well as the product line that you are working within?

How might targeted workers identify with certain new aesthetic directions in the context of their own working lives?

How might certain interactions with iconoclastic onscreen aesthetics promote emotional responses that are conducive to attentive, focused thinking?

What risks could be involved when breaking the mold in certain ways? Which of your iconoclastic ideas might be too inefficient or edgy, given that you are striving to create a highly functional tool for thinking work?

How might your team gain an understanding of whether targeted individuals see certain iconoclastic design concepts as being something new, appealing, and genuinely useful?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

M.

Planning Connection with Use

Valued computing tools are born from intensive conversations, and those conversations may then continue to evolve throughout a product's dispersion and adoption.

Designing for such meaningful connection requires critical thinking about potential real world scenarios of use — both desirable and negative — as well as potential interventions that might help steer usage toward intended outcomes.

During *application envisioning*, product teams can actively talk about potential downstream effects of their design concepts. Teams can also generate ideas about future connections with their applications' eventual users, envisioning integral touch points that can allow them to remain systemically responsive and strategically relevant over time.

Beyond input gathered from targeted individuals during product development, a launch date can represent the beginning of a meaningful dialog between a computing tool's creators and its users. As industries, cultures of practice, and technological environments evolve, so must the interactive applications that are situated within them. Conversations around adopted usage can provide a wellspring of insights for technologists striving to provide enduring value in complex work practices.

Long before a computing tool's launch, product teams can envision connections with real world use in order to develop design strategies that could positively influence the socio-technical systems that will envelop their creations. For example, envisioned concepts could contain integral channels for ongoing collaboration around user needs and design advancements. Teams may even go so far as to conceptualize their entire offerings as services instead of products, either literally or in spirit. Workers can experience vendor organizations through a variety of supportive, service oriented touch points that are thoughtfully tied into their activity contexts. The nature of these touch points may depend on the domain, the character of targeted work practices, and the level of disruptiveness or maturity of the product category.

Beyond connections with individual workers and organizations, product teams can also foster and learn from the communities of practice that will ideally grow around their applications. Once a product has been incorporated broadly into a knowledge work domain, workers in related professions may become invested in the tool's advancement. From users' perspectives, applications that succeed in becoming part of the infrastructure of a field can actually "belong," in some sense, to the people who have made the technology meaningful in their own practices.

This category contains 4 of the 100 *application envisioning* ideas in this book:

M1. Iterative conversations with knowledge workers

M2. System champions

M3. Application user communities

M4. Unanticipated uses of technology

Product teams can use these ideas to explore concepts for how their organizations and offerings could systemically support work practice throughout a product's evolution, based on mutually beneficial exchanges with users and stakeholders. Early ideation focused on this support, rather than post hoc efforts after a product launch, can help teams to integrate these relationships into the core of their application concepts. Early thinking about these connections can also positively shape the ongoing product development processes that an envisioning effort initiates.

The central notion of this category is most closely related to the "Exploring work mediation and determining scope" (A) and "Promoting integration into work practice" (K) categories.

M1. Iterative Conversations with Knowledge Workers

Product teams can iteratively co-envision valuable interactive applications with selected knowledge workers, grounding resulting technologies in current and emerging needs within targeted organizations and communities of practice. This dialog can commence in early, strategic design concepting and then continue throughout development and across product versions.

Examples from three knowledge work domains:

> An architect believes that the building modeling application that her studio uses needs some key improvements. After voicing her opinions to the vendor firm that created the computing tool, she is invited to provide feedback on early design concepts and prototypes for new and improved functionalities **(see illustration on next page)**.
>
> A financial trader uses a phone number in his trading application to contact a product team directly. He wants to let them know about some changes in regulations that will definitely impact the utility of certain functional options in their tool.
>
> A scientist opens her clinical research lab to a visiting product team. This team is meeting a sampling of their customers in order to gain a better understanding of how scientists are incorporating their own, and other vendors', analysis applications into complex laboratory processes.

Truly engaging, useful, and usable interactive applications may take multiple iterations to emerge. Unsurprisingly then, relatively frequent updates have become an assumed and essential part of many onscreen products' lifecycles.

To ensure that they are heading in desired and desirable directions, product teams can have authentic, ongoing conversations with targeted knowledge workers during any or all phases of their development processes. During *application envisioning*, teams can engage in conversations with potential users to identify where computing tools could provide value in their practices (A). These early conversations can also allow teams to gather input on their sketched application concepts, input which they can selectively and intelligently use to shape their products' essential forms.

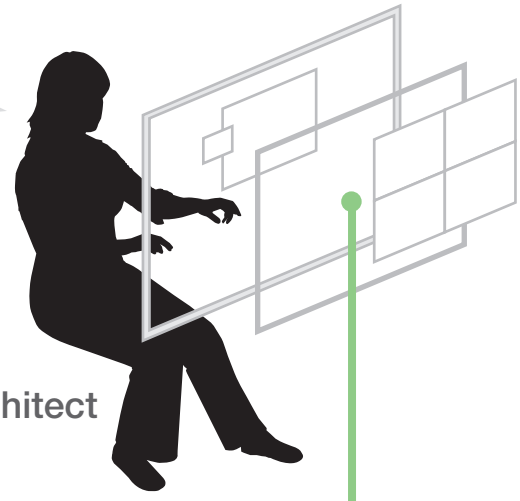
With the goal of an ongoing dialog in mind, teams can envision functionality concepts that could make it easier for their eventual users to establish contact or simply make suggestions (J1, J6). Workers may experience these channels as key touch points with a vendor, connectively extending outward from the tool itself into a larger, service oriented system.

When product teams do not actively consider an approach for iterative conversations with targeted knowledge workers, they may miss key opportunities while at the same time investing their efforts in other, nominally useful design strategies and functionality offerings (A9). This disconnection may damage brand and open the door to competing firms who make the effort to meaningfully engage in these conversations within targeted markets (K12).

Conversely, when teams work literally from workers' inputs instead of thoughtfully extracting the underlying intents of their comments, these ongoing conversations may result in unfocused and uncompelling "Frankenstein" applications (C).

See also: D1, G7, M

The vendor that created our building modeling application has been very interested to hear my thoughts on their new designs for the tool...



Architect

I'm often on the phone providing feedback to people from their team about simple prototypes that they put together to express their new ideas...

Vendor Product Team

Iterative research, concepting, design, and implementation based on ongoing conversation with carefully selected knowledge workers

And it's great to see those ideas come to life in new releases of their products, knowing that our firm's input made a difference in how they work...

Key *application envisioning* questions:

How might your team gather and use input from targeted knowledge workers as part of your *application envisioning* process? What functional channels within your product might allow you to gather such input over time? How could representative workers' insights, ideas, and feedback inform your decision making processes as you evolve your product?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Who should your team include in iterative conversations about your team's computing tool? What sampling of voices could represent key variabilities in your targeted markets — including so called "lead users" who often advance their own, local innovations?

What larger market trends could impact your ideas about having discussions with representative knowledge workers? Do targeted individuals and organizations expect to have personal interaction with their vendors?

How might your team get started with iterative conversations during early envisioning, before de facto choices appear in your application concepts?

Could informing participants become full time advisors in your team, or part of a larger network of confirmed but occasional reviewers?

What insights might these participants have into where technologies could provide new sources of value in their work practices?

What ideas might they have regarding desirable changes in how onscreen tools could fit into their activities?

What feedback might they provide about your sketched application and functionality concepts?

What processes could your team create to extract intents and prioritize knowledge workers' inputs, rather than taking them literally, without any filtering?

How might distilled inputs from participating informants impact the design strategy of your computing tool?

How could thinking of your offering as a service rather than a product change your team's perspective on knowledge workers' strategic inputs?

What functionality concepts might you envision to promote mutually valuable connections with your tool's expanding populations of end users? How might you gather input through these channels as you evolve your product to meet emerging needs over time?

What implications could such conversations have on the brand of your offering? On your marketing methods and messaging?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

M2. System Champions

Product teams can envision valuable support for individuals who champion the adoption and effective use of their interactive applications within certain communities of practice. These champions can be identified both within targeted customer organizations and within knowledge work fields at large.

Examples from three knowledge work domains:

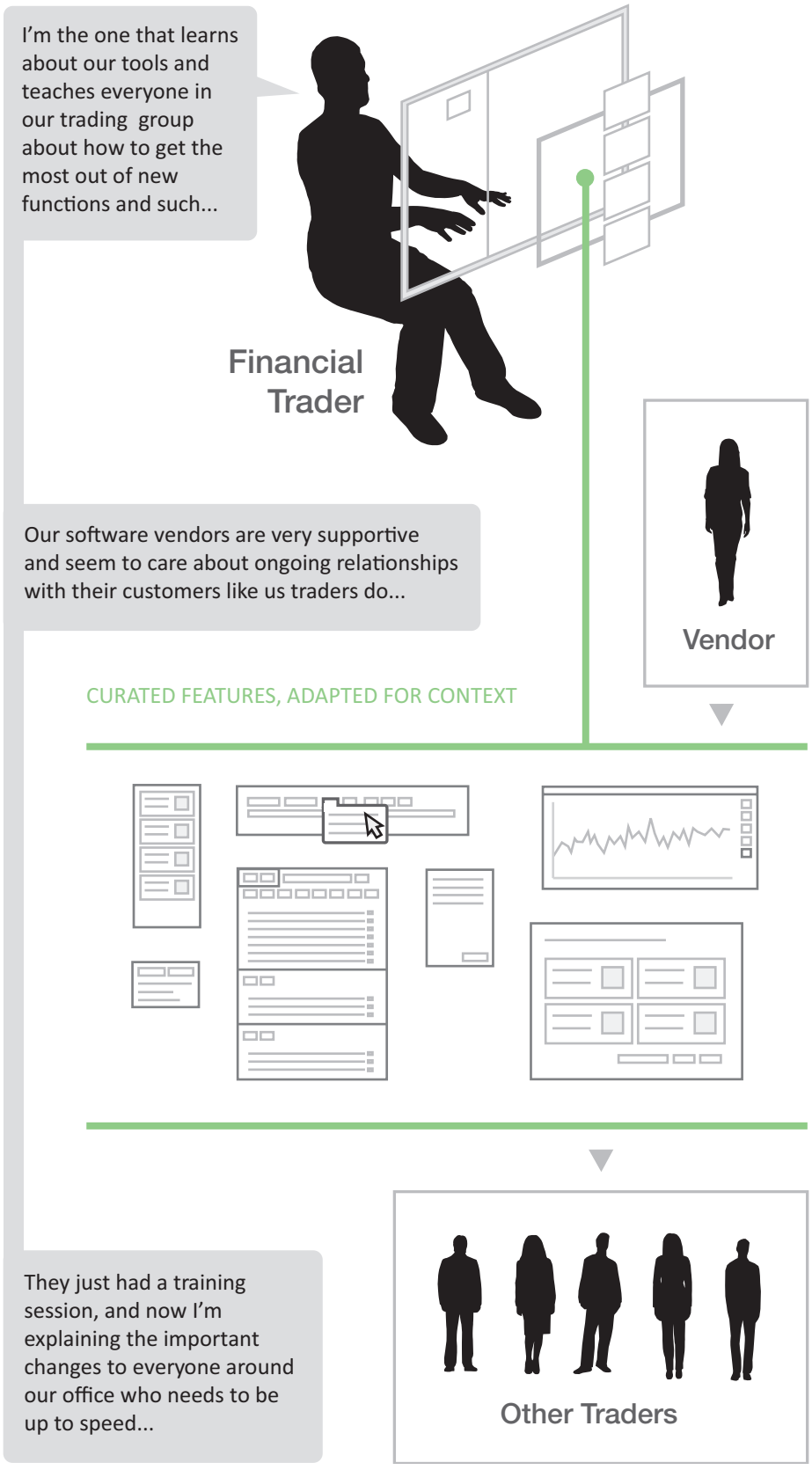
> A financial trader receives training about new trading application functionalities from a trusted vendor. He learns everything he can about changes in the current version so that he can effectively train his colleagues about the most relevant new features for their own ways of working **(see illustration on next page)**.
>
> A scientist starts a new position at a large clinical research lab. As she arrives at the new lab, she promotes the purchase and use of an analysis application that she found to be immensely valuable in her earlier efforts, ensuring that everyone understands its value and workings.
>
> An architect contacts a product vendor to ask how their building modeling application could be integrated with another computing tool that her studio is considering using. She is provided with a direct contact in the product's development team that can answer her detailed questions, and as a result of those personal conversations, she becomes an outspoken advocate of the modeling application within her firm and community.

Knowledge workers can become system champions through job responsibilities or through intrinsic interests and skills that make them "go to" people for questions about computing tools. These champions can act as translators between product teams and an application's other end users, reframing a tool's own "language" in the context of local processes and practices (A1, A7, A8). Champions may train other workers (K2, K7) and represent them in vendor relationships. Perhaps more importantly, such champions of an onscreen application often become the informal "help desk" that keeps mediated activities moving smoothly and effectively.

Product teams cannot "create" system champions, but they can promote the idea within their targeted markets and then watch for emerging individuals that could play the role. Not every customer organization will have a system champion, and some champions will not be associated with any particular organization. These voices can also arise in outside groups that contribute to larger fields and vocations, such as online communities (M3) and professional associations. Once teams have identified potential champions of their applications, product firms can supply these individuals with direct communication channels (J), specialized information, and targeted services to help them advance the adoption (K) and effective use of their computing tools.

When product teams do not actively consider support for system champions in their application concepts, strategic opportunities to scaffold real world use and gain valuable new sources of insights may remain overlooked (M1). More concretely, product support costs may be higher without well supported local champions responding to other workers' many complex yet day to day problems.

See also: A, D, G7, M


I'm the one that learns about our tools and teaches everyone in our trading group about how to get the most out of new functions and such...
Financial Trader
Our software vendors are very supportive and seem to care about ongoing relationships with their customers like us traders do...
Vendor
CURATED FEATURES, ADAPTED FOR CONTEXT
They just had a training session, and now I'm explaining the important changes to everyone around our office who needs to be up to speed...
Other Traders

Key *application envisioning* questions:

How might your team eventually identify and engage with system champions? What functionality concepts and interaction pathways could reach out to these targeted knowledge workers? What types of support could help them to effectively promote your computing tool in their own local environments and cultures of practice?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Who might be excited about the possibilities of applying your team's eventual product to real world work?

Which established roles in targeted organizations typically promote the adoption and use of new or updated computing tools?

Which existing users of your firm's products could be recognized as system champions?

Who is advancing the use of similar applications in your targeted markets, effectively "translating" them into local situations?

What processes might your team envision to ensure that your firm identifies and promotes relationships with emerging system champions?

What larger design and technology trends could influence your ideas about connecting with and supporting these potentially influential individuals?

What larger market trends could impact your team's ideas about connecting with those users who eventually champion your product?

How might system champions, in practice, actually advance the adoption and effective use of your computing tool? What background and instruction might they be called upon to provide? What types of problems could they face?

What functionality concepts might your team envision to provide targeted communication channels between system champions and supporting staff within your firm?

What additional specialized information about your product might system champions value?

What targeted services could your firm provide to support identified champions? What functionality concepts might your team sketch to enable and direct these service ideas?

How might system champions eventually represent their organizations, communities of practice, or larger professional fields in ongoing conversations with your team about the evolution of your product?

What implications could support for championing individuals have on the brand of your sketched computing tool? On your marketing methods and messaging?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

M3. Application User Communities

The social networks and collective focus of user communities can provide valuable support to knowledge workers who are trying to make the most of computing tools in their own organizations and personal practices. Product teams can envision concepts for fostering and reaching out to these communities, opening up channels to discuss issues and gather feedback.

Examples from three knowledge work domains:

> A scientist posts a question to an online forum about her analysis application and receives suggestions from other users, as well as a detailed answer from the vendor firm that created the tool **(see illustration on next page)**.
>
> An architect attends an annual convention where the user group for her building modeling application is hosting a panel discussion on different methods of applying the tool to real world projects. Since her firm is relatively new to using the product, she learns a lot from hearing about how other architectural studios approach their projects within it.
>
> A financial trader regularly meets with other traders working in the same market specialty. In addition to building business relationships, one of the usual topics of discussion is how to make better use of the industry standard applications that most firms use.

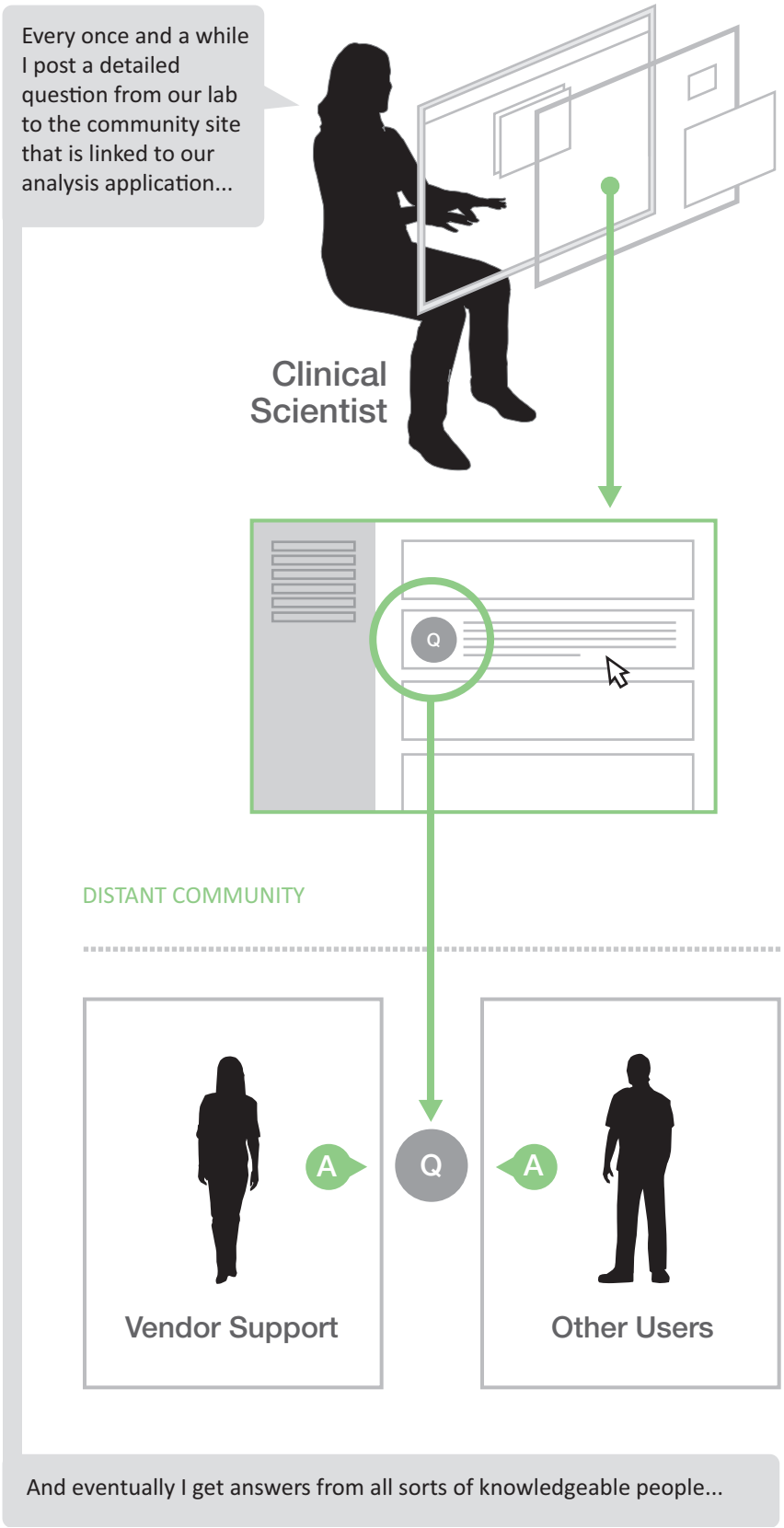
User communities can connect knowledge workers in a particular organization to a larger pool of people who are using computing tools in similar activities. Interactive applications with entrenched user bases can have large, active, and formalized user groups, whose members answer each other's questions and collectively lobby product firms. Alternately, domain specific communities, such as a group researching malaria cures, may discuss related computing applications as one part of a much larger conversation about their specialties.

Product teams can envision concepts for fostering the creation of user communities and supporting groups that arise organically. Once a community is established, teams can make responsive contributions, providing answers and technical support (M2), addressing concerns in a constructive manner, and soliciting input on new design concepts and prototypes.

Product firms and members of user communities can communicate through email lists (J1), centralized forums (J6), longer term knowledge bases (E1, G6, I7, J5), and face to face events. Workers may experience these channels as key touch points with a vendor, connectively extending outward from the tool itself into a larger, service oriented system. A user community's influence can extend into product strategy and development, brand reputation, purchasing behavior, the adoption process (K), and ongoing use (K12).

When product teams do not actively consider potential support for user communities in their application concepts, opportunities to scaffold adoption and gain new sources of valuable insights (M1) may remain overlooked. More concretely, application support costs may be higher without robust user communities responding to individual workers' many complex yet day to day problems.

See also: G3, G5, J2, M


Every once and a while I post a detailed question from our lab to the community site that is linked to our analysis application...
Clinical Scientist
Q
DISTANT COMMUNITY
A
Q
A
Vendor Support
Other Users
And eventually I get answers from all sorts of knowledgeable people...

Key *application envisioning* questions:

How could your firm be more than a "distant provider" to the larger communities that will eventually discuss and converge around your computing tool? What inputs might related communities contribute to your *application envisioning* efforts? How might interactive touchpoints and human support for certain communities eventually lead to positive impacts on product adoption, workers' outcomes, brand reputation, and other factors?

More specific questions for product teams to consider while envisioning applications for knowledge work:

How active are targeted individuals in various communities that are related to their professional practices?

Which existing, domain specific communities might be interested in mutually beneficial conversations about the formative direction of your product?

How might you connect with certain trusted and influential communities to gather insights, ideas, and feedback? What technologies and events do these groups congregate around and communicate through?

What larger design, technology, and market trends could impact your ideas about supporting user communities? Do contemporary collectives expect to have active conversations with related product vendors?

What new user communities might ideally form around the use of your envisioned product? How might the identities and segmentations of these groups reflect targeted domains and market segments?

How could your team foster the creation of one or more of these communities as part of actively releasing your technology?

What functionality concepts might your team envision to strongly tie the activities of certain user communities to your computing tool?

How could integral touch points, along with community action outside of your onscreen offerings, create opportunities for your firm to meaningfully connect with and support your user base?

How might your team's approaches for supporting application user communities relate to your other functionality concepts for supporting cooperation, collaboration, and workspace awareness?

What implications could connection with user communities have on the brand of your sketched application concepts? On your marketing methods and messaging?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

M4. Unanticipated Uses of Technology

History contains many examples of unanticipated uses that come to life once technologies are released into the world. Product teams can explicitly envision the design of their interactive applications to steer clear of support for certain usage scenarios. Teams can also inform the evolution of their offerings by investigating the unexpected ways that knowledge workers think about appropriating them.

Examples from three knowledge work domains:

> An architect uses tools in an airplane design application to create highly sculptural forms, which she then imports into her building modeling application to use as elements in an experimental building proposal **(see illustration on next page)**.

> A scientist searched extensively to find a data visualization tool for an ongoing, highly specific analysis need. Her clinical research lab now routinely uses a single interactive graph from an analysis tool that is normally only used by environmental engineers, not clinical researchers.

> A financial trader uses an instant messaging function, which was specifically designed as a method for rapidly booking deals with outside firms, to quickly distribute excess work to fellow traders in his group during peak intervals of activity.

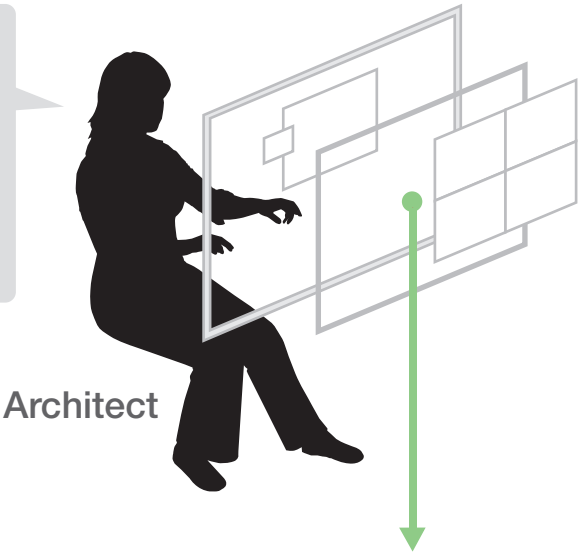
Although highly specialized interactive applications may somehow seem immune to unanticipated uses, knowledge workers often make use of computing products in unforeseen ways. Individuals and organizations commonly develop innovative new methods for applying a tool to the work practices that it was originally designed to mediate (A7, A8). In other cases, workers may find further uses for products that are entirely outside the scope of what a product team had initially envisioned (A9). These lateral jumps can lead to an application's involvement in a broader range of knowledge work efforts, potentially even in other fields and professions.

Since each unexpected use can present opportunities to tailor or extend an application's strategic and functional characteristics, product teams can recognize and envision support for viable cases (M3). Teams can uncover unanticipated uses in a variety of ways. Early on, they can conduct thought experiments to envision appropriated uses of their computing tools that they would like to avoid for ethical, legal, or strategic reasons. During product implementation, extensive prototype studies (J1, J6) and other forms of discussion with users (M1) may also reveal the emergence of unanticipated practices and markets. As part of planning post release discovery processes, teams may also seek out so called "lead users" known for generating innovative practices around their tools.

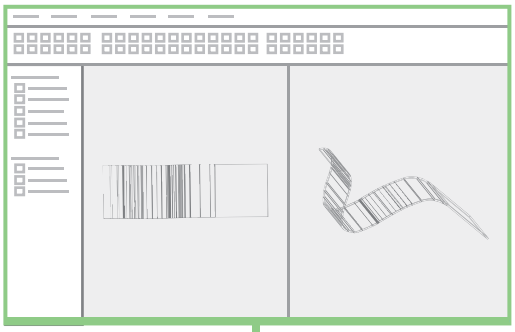
When product teams do not actively consider unexpected and emergent uses of their technologies, opportunities to harness workers' innovations (A, F) and to grow adoption in unexpected markets (K) can be lost. Technologists may also inadvertently create inherent opportunities for usage scenarios that are not consistent with their own intrinsic motivations and personal morals.

See also: B, C, D, G, M

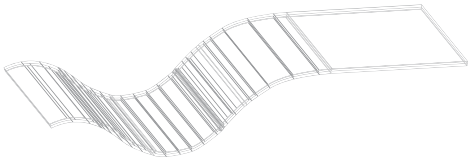
We heard that this leading architecture studio was using a different software package to come up with some of its more dramatic, organic forms...



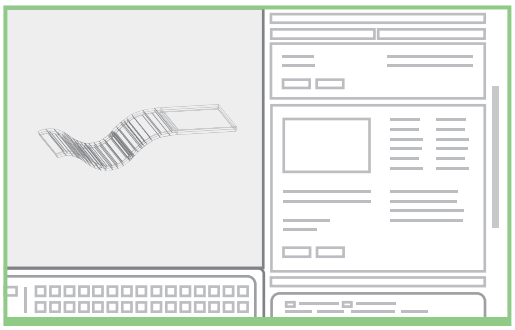
And so now we're also using this aviation software tool to do certain shapes. It generates them with much more info about the "how" of the structure, not just dumb 3D form data...



EXPORTED INFORMATION



We then import it into our main building modeling application, where it becomes part of our normal workflow...



Key *application envisioning* questions:

What early predictions might your team make about surprising and novel uses of your computing tool, simply by taking time to consider them? What inventive usages would you like to prevent or discourage due to ethical, legal, or strategic concerns? What processes might your team follow to identify emergent and unexpected uses of your product in a timely way?

More specific questions for product teams to consider while envisioning applications for knowledge work:

Are the professionals that your team is targeting known for deliberately using their tools in innovative ways, or do they seem more likely to adopt technologies as they were intended to be applied?

What technologies have targeted individuals and organizations appropriated in novel ways in the past? What might your team learn about your audience from these stories?

What larger design and technology trends could influence your ideas about unexpected scenarios of use?

What thought experiments might your team conduct to uncover potential unanticipated uses of your application concepts — while your product is still just abstract models and sketches?

What inventive usages might targeted knowledge workers predict?

What important variabilities in working methods might your team have overlooked when rationalizing work practice for design purposes?

What other, laterally related domains and occupations might reach out to some or all of the functionality in your application concepts?

Of the potential "not as designed" uses that your team has identified, which would be inconsistent with your own goals and morals? Which would clearly not fit the envisioned design strategy or brand of your application concepts?

How might your team "design out" certain usage scenarios that you do not want to promote or enable?

What plans might you establish for gathering information on novel uses of your product in its eventual user base? For working with innovative "lead users" to translate their related ideas and creations into valuable elements of your computing tool?

Do you have enough information to usefully answer these and other envisioning questions? What additional research, problem space models, and design concepting could valuably inform your team's *application envisioning* efforts?

Glossary

Many specialized terms have been intentionally omitted throughout "Working through Screens" in favor language that product teams can more easily share during their *application envisioning* efforts. However, given that the main goals of this book represent a highly specialized pursuit, the following glossary defines a number of specific terms that have been used in this volume. Please note that the following definitions are not general purpose; they are written specifically for the limited context of envisioning interactive applications for knowledge work. Broader definitions of the same terms could take on substantially different slants.

Activity: Within a product team's rationalized models of work practice, an activity is a larger set of goal oriented actions performed by one or more workers in order to contribute to a larger purpose. Activities are tied to foundational motives of knowledge work, and they are often nameable, discussed elements of workers' shared cultural models. As product teams envision concepts for how their technology could mediate workers' efforts, the selection of targeted activities can be a key determinant of an application's design strategy, functional scope, and potential meanings to future users. Activities are one part of the "operations, tasks, and activities" hierarchical approach to modeling work (see other definitions in this glossary), as coarsely borrowed from Alexei Leontiev's Activity Theory. Activities may also nest into other, higher order activities, which may in turn nest into further activities, and so on.

Advanced Analogies: Lateral references to the innovative use of technologies in other, often seemingly unrelated, domains. An advanced analogy creates a meaningful connection to an outside reference point that can inspire product teams to think about their design problems in different ways. Sometimes this inspiration involves literal translation from an analogous situation, while other times the forward looking influence is less direct and more evocative.

Analysis application: In the context of this book's examples, this term refers to a computing tool for clinical research. Analysis tools designed for the scientific market represent some of the most advanced examples of interactive applications currently available to knowledge workers. These tools can take seemingly countless pieces of laboratory data and present them in ways that allow scientists to understand trends, uncover anomalies, and make decisions. Robust visualization functionality can allow researchers to sift through experimental results from a wide variety of perspectives based on emergent wayfinding approaches. In clinical research areas where certain established analyses are often useful for understanding data, highly tailored functions can automate known, well characterized tests and present their results in clear and actionable information displays. See the "Primer on example knowledge work domains" section for additional background information on the clinical research computing examples used throughout this book.

Application: See definition for "interactive application."

Application concept: Sketched assemblies that integrate selected functionality concepts, along with strategic and infrastructural ideas about a product's design, into a named, overall approach to mediating a chosen scope of knowledge work activities. Product teams eventually choose one of their envisioned application concepts (or a hybrid concept) to serve as the basis for the rest of the product development process.

Application envisioning: An early, separate time in product development for teams to collaboratively consider diverse and appropriate concepts of what could be, rather than being pulled down an incremental, largely unconsidered course. During this interval, teams can iteratively generate application concepts in lightweight models and sketches rather than implemented code. Time spent in this "pre-production" mode can result in outcomes that are grounded in the vector of a compelling and broadly informed design strategy. At the end of *application*

envisioning, teams can select the "fittest" concepts for their product's essential direction and form from a relevant ecosystem of potential ideas.

Architect: See definition for "architectural domain." See also the "Primer on example knowledge work domains" section for additional background information.

Architectural domain: Architects and their firms, generally speaking, seek to profitably create well designed drawings for buildings that address complex criteria. Some aspects of these complex work practices are used in examples throughout this book. See the "Primer on example knowledge work domains" section for additional background information.

Automaticity: In the context of interactive applications for knowledge work, the point of learning and accommodation at which workers can execute certain operations and larger tasks with reduced effort. When novel situations arise in otherwise predictable interactions, workers' automatic behaviors can lead to special error cases in product use. Application designs that promote automaticity in frequent usage scenarios can be very different from those design responses that emphasize initial learnability and usability.

Building modeling application: This term encompasses an emerging class of computing applications, more commonly called Building Information Modeling (BIM), that is beginning to drive radical changes in architectural practice. In BIM, the entire design of a building is stored as a collaborative virtual model that can be modified and referenced by different contributors to a project, purportedly improving communication and reducing representational misunderstandings. See the "Primer on example knowledge work domains" section for additional background information on the architectural computing examples used throughout this book.

Clinical research domain: Clinical research scientists, generally speaking, want to make applied discoveries related to human health. These scientists adopt diverse methods and technologies to attack their research problems, depending on the nature of the topic under study and researchers' own areas of expertise. Some aspects of these complex work practices are used in examples throughout this book. See the "Primer on example knowledge work domains" section for additional background information.

Community of practice: A community of knowledge workers mutually engaged in collaborative action and learning in a shared domain over time. As described in the writings of Etienne Wenger, these communities build upon each others' efforts and ideas, often through direct participation in social exchanges. In the context of knowledge work, communities of practice can represent the staff of a particular group in an organization or a larger collective within a domain, such as a long standing professional organization.

Conceptual model: In the context of envisioning applications for knowledge work, they are the mental models that contain people's particular understandings of which work practices an interactive application is designed to support, how it essentially "works," and how it might fit into their own activities. As mental constructs, product teams cannot create conceptual models for their eventual users. Knowledge workers must build their own nesting and interrelated understandings as part of adopting a computing tool into their own work practices. However, product teams can develop intended conceptual models for various parts of their applications and seek to communicate these models through embedded design characteristics and explicit instruction.

Domain: Generally connotes a unique specialty in knowledge work. Domains can encompass a broad range of existing cultural knowledge and skills, ranging from general methods of practicing to highly specific information and abilities that are crucial for successfully accomplishing work activities.

Design strategy: The singular, relatively unchang-

ing proposals that summarize the essence of an envisioned application's scope, core value, points of emotional connection, and approaches to mediating knowledge work. Design strategies are situated within a larger context of targeted user needs, technological possibilities, market forces, trends, and predictions. Product teams can use these strategies to drive clarity in their offerings and focus their members around a shared vision and goal set. Since they are derived from key business, marketing, and product development considerations, design strategies can be thought of as a lower level expression of a computing tool's initiating, high level charter.

Envisioning: See definition for "application envisioning."

Envisioning questions: Points of inquiry that get product teams thinking about divergent factors that could shape their application concepts. Envisioning questions are answered through active, considered, and participatory processes of research and design exploration.

Financial trader: See definition for "financial trading domain." See also the "Primer on example knowledge work domains" section for additional background information.

Financial trading domain: The many specializations of financial trading are, generally speaking, about the exchange of financial instruments to maximize returns for traders, their firms, and their clients. Some aspects of these complex work practices are used in examples throughout this book. See the "Primer on example knowledge work domains" section for additional background information.

Functional area: A larger area within an application, often organized around a particular goal or set of actions. Functional areas can be designated by a prominent heading in an application's navigation and a separate "place" or "section" within a tool. A single functional area can typically be further broken down into a number of individual options and related interactivities. See also definitions for "functionality" and "functionality concept."

Functionality: Features in a knowledge work application that are provided as goal oriented options for workers to use in their own activity contexts. Some functionalities may be highly specific tools for narrow goals, while others may be more general purpose and open to users' interpretations. Applications are typically comprised of many functionalities, and individuals may use some available options more frequently than others, depending on how they see a product fitting into their local ways of thinking and acting. See also definitions for "functional area" and "functionality concept."

Functionality Concept: Sketched possibilities for work mediation that product teams generate as a part of their envisioning explorations. These concepts represent a goal oriented scenario for potential user experience, including a team's proposed design responses to targeted problems and constraints. See also definitions for "functional area" and "functionality."

Information management application: In the context of this book's examples, this term refers to a computing tool for clinical research. These applications, also known as a Laboratory Information Management Systems (LIMS), can keep track of all stored data about a laboratory, from the stock on the shelves to the results of genetic tests. Many of these systems also provide functionality for defining and monitoring laboratory workflow, allowing scientists to design and distribute experimental protocols for lab technicians and automated instruments to follow. Since LIMS are often open to integration with other applications, they can become a central hub for connecting all of a laboratory's computing infrastructure. See the "Primer on example knowledge work domains" section for additional background information on the clinical research computing examples used throughout this book.

Interaction object: The comprehensible elements

within an application that workers act on in order to accomplish their goals. Interaction objects are defined from users' activity oriented perspectives, and they do not necessarily correspond to the idea of "objects" in the computer science sense of the word. Product teams can translate key interaction objects from artifacts in workers' existing practices, which can then become meaningful elements of an application's intended conceptual models. Other objects can represent novel system ideas that would otherwise not exist in targeted knowledge work culture. Interaction objects can nest and interrelate, and ideas for additional, low level objects may emerge during the product development process. The notion of interaction objects is derived from Ben Shneiderman's "Object-Action Interface Model" (1998), without its emphasis on direct manipulation.

Interaction pattern: A reusable convention in the design of interactive applications. These conventions are often thought of at the "literal" level, containing specific arrangements of information and user interface controls. Interaction patterns can also usefully represent larger commonalities in conventional interactions, such as approaches to entire application structures or task processes.

Interactive computing: In the context of knowledge work, applications of computing that workers directly interact with, as opposed to an increasing number of embedded applications that are effectively hidden in artifacts in a way that prevents users from having direct control over their behaviors. This book primarily focuses on interactive applications that could be feasibly implemented on personal computers at the time of writing.

Interactive application: A specialized computing tool for mediating targeted activities, including both installed products and those that are accessed via the Internet. In the context of envisioning potential user experiences in knowledge work, the emphasis of this term falls less on the technical construction of a tool and more on its potential definition and design opportunities.

Interaction model: The highest level expression of an application's structure. A "shell" that determines how interactive behaviors and disclosures essentially flow within a computing tool. An interaction model frames the full scope of an onscreen product, outlining the interactive approaches and points of access for its constituent functional concepts. In contemporary personal computing, where some sort of keyboard and pointing method are essentially a given, interaction models may reside largely in the onscreen structure and behaviors of a knowledge work tool, rather than in specialized hardware controls.

Knowledge work: A broad category of human activity that is focused on inventing, producing, interpreting, manipulating, transforming, applying, and communicating information using specialized skills and knowledge.

Knowledge workers: Individuals who, in their working lives, are valued for their specialized intellectual skills and their ability to act on and with complex information in goal oriented ways. This term can refer to specialized professions (such as the architect, financial trader, and scientist found in the examples throughout this book) or more generalized vocations.

Object: See definition for "interaction object."

Organization: Any group of individuals working together with shared goals and, in many cases, a division of labor and responsibilities. This umbrella includes groups such as non profits and online collaboratives, as well as those that are more commonly associated with knowledge work in the business world.

Magic happens expectation: A problematic expectation in product teams that a successful, even visionary, product will somehow emerge solely from the sum of countless detailed definition, design, and implementation decisions — without a larger design strategy or application concept as guiding road map. See also definitions for "straight to the details progression" and "single vision and concept

design."

Market information application: These computing tools allow financial traders to investigate current and historical data about specific market sectors and traded financial instruments from a variety of different perspectives. The feeds and visualizations in these applications can help traders to stay current on market happenings and to better assess potential deals. See the "Primer on example knowledge work domains" section for additional background information on the financial trading computing examples used throughout this book.

Mediate / mediation: Refers to an interactive application's interfacing role between workers and their goals, as coarsely borrowed from Alexei Leontiev's Activity Theory. Each approach to supporting a work practice with technology presents different "mediating" changes to the essential nature of that practice. Knowledge workers will inevitably view some mediation approaches as more attractive and successful than others.

Models of problem spaces: Artifacts that summarize meaningful primary research, secondary research, and design research insights into clear and informative representations that map out relevant regions of product possibility. These models can take a variety of forms, ranging from graphs, to textual stories, to storyboards, to video exposition.

Offloading: Reducing the work needed to accomplish an action by distributing some of the effort to an interactive application, collaborator, or another part of a distributed work system. Actions that initiate offloading can range from deliberate and intentional to implicit and subconscious. Offloading effort can change the essential nature of work practices. After a computing tool has been appropriated into a workplace culture, participating individuals, in their accommodated ways of thinking and acting, may not even recognize how they opportunistically offload effort to the external artifact.

Operation: Within a product team's rationalized models of work practice, operations are low level actions that workers can perform. Individual operations are typically enacted in a sequence in order to accomplish larger goals for interacting with a system. Operations are one part of the "operations, tasks, and activities" hierarchical approach to modeling work (see other definitions in this glossary), as coarsely borrowed from Alexei Leontiev's Activity Theory. Multiple operations comprise a task.

Product: See definition for "application." In many cases, the term "service" could be equally applicable (though it was not extensively used throughout this book in order to promote brevity). Although the term "product" has a commercial connotation, the applications discussed here could also be created by an open source community or internally developed within specific organizations.

Product Teams: The primary audience of this book; a group of people creating an interactive application for knowledge work. For the purposes of this text, product teams' memberships can include anyone who plays a role in product development, with a special emphasis on those individuals who are (or could be) involved during early, strategic, *application envisioning* efforts. This broad definition can include anyone from high level management to knowledge work "customers" who are brought on as advisors and design participants.

Progressive disclosure: A design approach which can decrease perceptual load and promote more effective use of limited screen areas by "hiding" some content and functionality until it is interactively accessed, as needed, through users' goal oriented pathways.

Scaffolding: Borrowed from Lev Vygotsky's ideas on the Zone of Proximal Development, scaffolding is application content or functionalities that provide workers with supporting structures for their learning, based on an understanding of their current knowledge and abilities. Users can accomplish more with the support of scaffolding, and by doing so, they can make learning leaps that may be more

difficult without such support. After an individual has learned a scaffolded interaction, the supporting features can often be removed from day to day use. Alternately, in some cases, these features may be left in place to provide some ongoing utility.

Scientist: See definition for "clinical research domain." See also the "Primer on example knowledge work domains" section for additional background information.

Settings: Meaningful parameters in an application that workers can have some measure of control over. These parameters can be highly flexible and numerous in applications for early adopters. By contrast, in highly developed products that target less invested user segmentations, guiding product settings may be relatively few in number and more constrained in scope.

Single vision and concept design: The problematic approach by which product teams iteratively create only one de facto design strategy and corresponding application concept. These singular progressions often begin with a rush toward implementation and only limited ideas about product direction and potential meaning. Although teams practicing this type of design may evolve their own conceptions about their applications at a detailed level, their lack of early, divergent thinking about work mediation can be considered a failure to strategically explore potential directions within their initial, high level charters. See also "straight to the details progression" and "magic happens expectation."

Sketch: A rough representation of a framing idea or potential design direction, generated within a product team during the *application envisioning* process. Teams can sketch at many levels of granularity, ranging from an application's overall vector to the diminutive shape of a small functionality concept. During early envisioning, teams typically create sketches to capture and convey potential options for mediating work, not to solidify highly specific design details like those found in later prototypes. Although the outputs of sketching exercises are often some sort of drawing, teams may also usefully "sketch" their ideas in video and other media.

Straight to the details progression: The problematic progression by which product teams quickly move from high level consideration of product strategy into the specifics of an application's definition, design, and implementation. This progression can be fueled by various team members' specialized focuses on particular techno-centric facets of their products. See also "magic happens expectation" and "single vision and concept design."

Tailored: Designed or customized to meet the specific needs of targeted knowledge workers and their organizations, as situated in the context of well characterized cultures and activities.

Task: Within a product team's rationalized models of work practice, tasks are a goal oriented action that workers can perform. Tasks may be nameable, discussed elements of workers' shared cultural models of their own work, or they may be unspoken routines and improvisations. A product team's application concepts may aim to effectively eliminate some existing tasks in workers' practices while introducing new ones that could stem from the adoption of their product. Tasks are one part of the "operations, tasks, and activities" hierarchical approach to modeling work (see other definitions in this glossary), as coarsely borrowed from Alexei Leontiev's Activity Theory (where the term "Action" is used instead of "Task"). Multiple operations comprise a task. Multiple tasks comprise an activity, though not all of the tasks associated with a particular activity may occur within the confines of a single interactive application.

Trading application: As a financial traders' "window" into their firm's valuable information, these computing tools typically allow users to examine shared trading records and to book new deals. Trading applications often automate many tedious operations ranging from small data predictions when filling out forms to automatic routing and

processing of completed deals through a number of related systems. See the "Primer on example knowledge work domains" section for additional background information on the financial trading computing examples used throughout this book.

Work practice: The actual methods, as opposed to idealized notions of process, by which knowledge workers accomplish their efforts. People may have different approaches to accomplishing the same categories of activity, and observations of knowledge work often reveal considerably more improvisation, situated decision making, communication, cooperation, and collaboration than is commonly acknowledged by individual workers or their organizations. See also definition for "work process."

Work process: Established ways of working that are formally agreed upon within an organization or larger profession. In many types of knowledge work, especially in those domains where practitioners have developed considerable skills and expertise, established processes may arise from formal acknowledgement and standardization of workers' own emergent practices. Alternately, work processes may be defined based on top down considerations, such as legal or business operations requirements. See also definition for "work practice."

Workflow: Established work processes where efforts are distributed among a number of different individuals, often based on the assigned roles that these people play within a group or larger organization. Depending on the context, workflow can also be a synonym for "work process."

Workspace awareness: As outlined in Gutwin and Greenberg (2002), this type of awareness can be thought of as an ongoing sense, often without conscious attention, of others' actions within a shared locale. In shared computing environments, this awareness can be especially relevant when it pertains to actions that could impact cooperative or collaborative efforts. Workspace awareness can be critical for promoting coordination and communication, potentially leading to fewer errors and higher quality work outcomes. Knowledge work applications can provide specialized workspace awareness functionalities to counter the individualized, often isolating interactions of contemporary personal computing.

Bibliography

Abel, C. 2004. Architecture, Technology and Process. Burlington: Architectural Press / Elsevier.

Alexander, C. 1964. Notes on the Synthesis of Form. Cambridge, Massachusetts: Harvard University Press.

Alvesson, M. 2004. Knowledge Work and Knowledge-Intensive Firms. Oxford: Oxford University Press, USA.

Antonelli, P. (ed.). 2002. Workspheres: Design and Contemporary Work Styles. New York: The Museum of Modern Art, New York.

Auger, B. 1972. The Architect and the Computer. Westport, Connecticut: Praeger Publishers.

Baird, D. 2004. Thing Knowledge: A Philosophy of Scientific Instruments. Berkley: University of California Press.

Beyer, H., and K. Holtzblatt. 1998. Contextual Design: Defining Customer-Centered Systems. San Francisco: Morgan Kaufmann Publishers Inc.

Berg, M. 1997. Rationalizing Medical Work: Decision Support Techniques and Medical Practices. Cambridge, Massachusetts: The MIT Press.

Bergeron, B. 2002. Bioinformatics Computing. Upper Saddle River, New Jersey: Prentice Hall PTR.

Bodker, K., and F. Kensing. 2004. Participatory IT Design: Designing for Business and Workplace Realities. Cambridge, Massachusetts: The MIT Press.

Borchers, J. 2001. A Pattern Approach to Interaction Design. Chichester, West Sussex: John Wiley & Sons Ltd.

Brown, J. S., and P. Duguid. 2002. The Social Life of Information. Cambridge, Massachusetts: Harvard Business School Press.

Brown, R., and J. Kulik. 1977. Flashbulb memories. Cognition 5, 73–99.

Bush, V. 1945. As we may think. The Atlantic Monthly 7: 101-108.

Buxton. B. 2006. Lecture: Sketching and experience design. November Boston CHI meeting held at Sun Microsystems in Burlington, Massachusetts.http://www.brightcove.com/title.jsp?title=323680309&channel=324389485.

Buxton, B. 2007. Sketching User Experiences: Getting the Design Right and The Right Design. San Francisco: Morgan Kaufmann.

Campbell-Kelly, M. 2004. From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry. Cambridge, Massachusetts: The MIT Press.

Cara, F. 1999. Cognitive Ergonomics. In Wilson, R. A., and Keil, F. C. (eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Massachusetts: The MIT Press.

Card, S. K., J. Mackinlay, B. Shneiderman (eds.). 1999. Readings in Information Visualization: Using Vision to Think. San Francisco: Morgan Kaufmann.

Carroll, J. M. (ed.). 2003. HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science. San Francisco: Morgan Kaufmann Publishers.

Causton, H., J. Quackenbush, and A. Brazma. 2003. Microarray Gene Expression Data Analysis: A Beginner's Guide. Malden, Massachusetts: Blackwell Publishing Limited.

Chang, W. C., and T. Y. Wu. 2007. Exploring types and characteristics of product forms. International Journal of Design 1(1): 3-14.

Chi, E. H., P. Pirolli, K. Chen, and J. Pitkow. 2001. Using information scent to model user information needs and actions and the Web information scent. Proceedings of ACM CHI 2001 Conference on Human Factors in Computing Systems, pp. 490-497.

Clark, A. 2004. Natural-Born Cyborgs: Minds, Tech-

nologies, and the Future of Human Intelligence. Oxford: Oxford University Press, USA.

Cooper, A. 1995. About Face: The Essentials of User Interface Design. Foster City: IDG Books.

Cooper, A. 1999. The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity. Boston: Pearson Education.

Csikszentmihalyi, M. 1991. Flow: The Psychology of Optimal Experience. New York: Harper Perennial.

Davenport, T. H. 1992, Process Innovation: Reengineering Work through Information Technology. New York: McGraw-Hill Companies.

Davenport, T. H. 2005. Thinking for a Living: How to Get Better Performances and Results from Knowledge Workers. Cambridge, Massachusetts: Harvard Business School Press.

Demetrios, E. 2002. An Eames Primer. New York: Universe Publishing.

Dollens, D. 2002. D2A: Digital to Analog Architecture. Santa Fe: Lumen Books/Sites Books.

Dourish, P. 2001. Where the Action Is: The Foundations of Embodied Interaction. Cambridge, Massachusetts: MIT Press.

Duncan, J. 1999. Attention. In Wilson, R. A., and Keil, F. C. (eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Massachusetts: The MIT Press.

Dunne, A., and F. Raby. 2001. Design Noir: The Secret Life of Electronic Objects. Basel: Birkhäuser.

Englebart, D. Augmenting Human Intellect: A Conceptual Framework. In Packer, R., and K. Jordan. (eds.). 2002. Multimedia: From Wagner to Virtual Reality, Expanded Edition. New York: W. W. Norton & Company.

Fayyad, U., G. Grinstein, and A, Wierse. 2001. Information Visualization in Data Mining and Knowledge Discovery. San Francisco: Morgan Kaufmann.

Fitzpatrick, G. 1998. The Locales Framework: Understanding and Designing for Cooperative Work. Doctoral dissertation, University of Queensland, Brisbane.

Florida, R. 2003. The Rise of the Creative Class: And How It's Transforming Work, Leisure, Community and Everyday Life. New York: Basic Books.

Fry, B. 2004. Computational Information Design. Doctoral dissertation, MIT Program in Media Arts and Sciences, School of Architecture and Planning. http://acg.media.mit.edu/people/fry/phd/dissertation-050312b-acrobat.pdf. Accessed Decemeber 2007.

Fulton Suri, J. 2005. Thoughtless Acts?: Observations on Intuitive Design. Watertown, Massachusetts: Chronicle Books.

Gentner, D.R., and J. Grudin. 1996. Design models for computer-human interfaces. IEEE Computer 29 (6), 28-35.

Gorman, C. 2003. The Industrial Design Reader. New York: Allworth Press.

Grudin, J. 1990. The computer reaches out: The historical continuity of interface design. CHI '90 Proceedings, pp. 261-268.

Grudin, J. 1994. Computer-supported cooperative Work: History and focus. IEEE Computer 27 (5), 19-26.

Gutwin, C., and S. Greenberg. 2002. A descriptive framework of workspace awareness for real-time groupware. Computer Supported Cooperative Work, 11(3-4), 411-446.

Gutwin, C., S. Greenberg, R. Blum, and J. Dyck. 2005. Supporting informal collaboration in shared workspace groupware. HCI Technical Report 2005-

01, The Interactions Lab, University of Saskatchewan, Saskatoon, Saskatchewan, Canada.

Greenberg, S. and B, Johnson. 1997. Studying awareness in contact facilitation. Position paper for the ACM CHI'97 Workshop on Awareness in Collaborative Systems, organized by McDaniel, S. E. and Brinck, Atlanta, T., Georgia, March 22-27.

Hackos, J. T., and J. C. Redish. 1998. User and Task Analysis for Interface Design. Hoboken: John Wiley & Sons.

Harriss, H., S. Winstanley, and G. Vickers. (ed.). 2005. Capture It: A Future Workplace for the Multi-Generational Knowledge Worker. London: Helen Hamlyn Research Centre.

Henderson, K. 1998. On Line and On Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering. Cambridge, Massachusetts: The MIT Press.

Heufler, G. 2004. Design Basics: From Ideas to Products. Sulgen, Switzerland: Arthur Niggli.

Hirshberg, J. 1999. The Creative Priority: Putting Innovation to Work in Your Business. New York: Collins.

Hiltzik, M. A. 2000. Dealers of Lightning: Xerox PARC and the Dawn of the Computer Age. New York: Collins.

Hutchins, E. 1995. Cognition in the Wild. Cambridge, Massachusetts: The MIT Press.

Hutchins, E. 1999. Cognitive artifacts. In Wilson, R. A., and Keil, F. C. (eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Massachusetts: The MIT Press.

Kelley, T. 2001. The Art of Innovation: Lessons in Creativity from IDEO, America's Leading Design Firm. New York: Doubleday.

Kurosu, M., and K, Kashimura. 1995. Apparent usability vs. inherent usability: experimental analysis on the determinants of the apparent usability. Conference companion on Human factors in computing systems. 292-293. Denver, Colorado. May 7-11.

Laurel, B. 2003. Design Research: Methods and Perspectives. Cambridge, Massachusetts: The MIT Press.

Landauer, T. K. 1996. The Trouble with Computers: Usefulness, Usability, and Productivity. Cambridge, Massachusetts: MIT Press.

Lawson, B. 1997. How Designers Think. Burlington: Architectural Press / Elsevier.

Lawson, B. 2004. What Designers Know. Burlington: Architectural Press / Elsevier.

Lazar, J., A. Jones, K. Bessiere, I. Ceaparu, B. Shneiderman. April 2004. User frustration with technology in the workplace. HCIL Laboratory publication: HCIL-2004-12, CS-TR-4662, ISR-TR-2005-53.

Lee, B. 2006. From Cognitive Artifacts to Social Artifacts: The iDeas Design Ecology. Proceedings of UIST '06, October 15–18, 2006, Montreux, Switzerland.

Lidwell, W., K. Holden, and J. Butler. 2003. Universal Principles of Design: 100 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design. Gloucester, Massachusetts: Rockport.

Live|Work. Service Design Glossary. http://www.livework.co.uk/glossary/. Accessed December 2007.

Lowgren, J., and E. Stolterman. 2004. Thoughtful Interaction Design: A Design Perspective on Information Technology. Cambridge, Massachusetts: The MIT Press.

Maeda, J. 2006. The Laws of Simplicity. Cambridge, Massachusetts: The MIT Press.

Microsoft Corp., Windows User Interface Technical Articles, Microsoft Inductive User Interface Guidelines, http://msdn2.microsoft.com/en-us/library/ms997506.aspx. February 9, 2001. Accessed December 2007.

Mijksenaar, P. 1997. Visual Function: An Introduction to Information Design. Princeton: Princeton Architectural Press.

Mirel, B. 2003. Interaction Design for Complex Problem Solving: Developing Useful and Usable Software. San Francisco: Morgan Kaufmann.

Moggridge, B. 2006. Designing Interactions. Cambridge, Massachusetts; MIT Press.

Nardi, B. A. 1993. A Small Matter of Programming: Perspectives on End User Computing. Cambridge, Massachusetts: MIT Press.

Nardi, B. A. 1996. Context and Consciousness: Activity Theory and Human-Computer Interaction. Cambridge, Mass: MIT Press.

Nardi, B. A., and V. O'Day. 1999. Information Ecologies: Using Technology with Heart. Cambridge, Massachusetts: MIT Press.

Nemeth, C., M. O'Connor, P. A. Klock, R. Cook. 2005. Cognitive artifacts' implications for health care information technology: Revealing how practitioners create and share their understanding of daily work. Advances in Patient Safety: Vol. 2 AHRQ Publication No. 05-0021-2. http://www.ctlab.org/documents/Cognitive%20Artifacts%20Implications.pdf

Nielsen, J., and R. L. Mack. 1994. Usability Inspection Methods. Hoboken: John Wiley & Sons.

Norman, D. A. 1990. The Design of Everyday Things. New York: Doubleday.

Norman, D. A. 1990. The 'problem' of automation: Inappropriate feedback and interaction, not 'over-automation.' Philosophical Transactions of the Royal Society of London B 327: 585–593.

Norman, D. A. 1993. Things That Make Us Smart: Defending Human Attributes in the Age of the Machine. Reading, Mass: Addison-Wesley Publishing.

Norman, D. A. 2001. Applying the Behavioral, Cognitive, and Social Sciences to Products. http://www.jnd.org/dn.mss/applying_the_behavio.html. Accessed December 2007.

Norman, D. A. 2004. Emotional Design: Why We Love (or Hate) Everyday Things. New York: Basic Books.

Office for Metropolitan Architecture, and M. Kubo (ed.). 2005. Office for Metropolitan Architecture: Seattle Public Library. Barcelona: Actar.

Orr, J. E. 1996. Talking about Machines: An Ethnography of a Modern Job. Itheca: ILR Press/Corbell University Press.

Packer, R., and K. Jordan. (eds.). 2002. Multimedia: From Wagner to Virtual Reality, Expanded Edition. New York: W. W. Norton & Company.

Pinelle, D., C. Gutwin, and S. Greenberg. 2003. Task Analysis for Groupware Usability Evaluation: Modeling Shared-Workspace Tasks with the Mechanics of Collaboration. ACM Transactions on Human Computer Interaction 10(4), 281-311.

Postrel, V. 2003. The Substance of Style: How the Rise of Aesthetic Value Is Remaking Commerce, Culture, and Consciousness. New York: HarperCollins.

Raskin, J. 2000. Humane Interface: New Directions for Designing Interactive Systems. Reading, Mass: Addison-Wesley Publishing.

Reiser, J., and N. Umemoto. 2006. Atlas of Novel Tectonics. Princeton: Princeton Architectural Press.

Rettig, M., and A. Goel. Designing for experience: Frameworks and project stories. Presentation for

UX Week conference, New York Chapter of the Interaction Design Association. http://www.marcrettig.com/uxweek/rettig&goel.uxWeek.8.25.05.pdf. Accessed Dec 2007.

Rheingold, H. 2000. Tools for Thought: The History and Future of Mind-Expanding Technology. Cambridge, Massachusetts: MIT Press.

Rudder Baker, L. 1999. Folk psychology. In Wilson, R. A., and Keil, F. C. (eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Massachusetts: The MIT Press.

Sanoff, H. 1990. Participatory Design: Theory and Techniques. Raleigh, NC: Bookmasters.

Sarter N.B., D. D. Woods, and C.E. Billings. 1997. Automation surprises. In Salvendy, G. (ed.). Handbook of Human Factors & Ergonomics, second edition. Hoboken: John Wiley & Sons.

Sellen, A. J. and, Harper R. H. R. 2003. The Myth of the Paperless Office. Cambridge, Massachusetts: The MIT Press.

Sharp, H., Y. Rogers, and J. Preece. 2007. Interaction Design: Beyond Human-Computer Interaction. Hoboken: John Wiley & Sons.

Schneider, W. 1999. Automaticity. In Wilson, R. A., and Keil, F. C. (eds.). The MIT Encyclopedia of the Cognitive Sciences. Cambridge, Massachusetts: The MIT Press.

Shneiderman, B. 1998. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Reading, Massachusetts: Addison Wesley.

Simon, H. A. 1996. The Sciences of the Artificial. Cambridge, Massachusetts: The MIT Press.

Spinuzzi, C. 2003. Tracing Genres through Organizations: A Sociocultural Approach to Information Design. Cambridge, Massachusetts: The MIT Press.

Spillers, F. 2003. Task analysis through cognitive archeology. D. Diaper and N. Stanton (eds.) The Handbook of Task Analysis for HCI. http://www.experiencedynamics.com/pdfs/published_works/cognitive_archeology.pdf

Spillers, F. 2004. Emotion as a cognitive artifact and the design implications for products that are perceived as pleasurable. http://www.experience-dynamics.com/pdfs/published_works/Spillers-EmotionDesign-Proceedings.pdf. Accessed December 2007.

Sterling, B., and L. Wild. 2005. Shaping Things. Cambridge, Massachusetts: The MIT Press.

Suchman, L. A. 1987. Plans and Situated Actions: The Problem of Human-Computer Communication. New York: Cambridge University Press.

Thackara, J. 2006. In the Bubble: Designing in a Complex World. Cambridge, Massachusetts: The MIT Press.

Tufte, E. R. 1983. The Visual Display of Quantitative Information. Cheshire, Connecticut: Graphics Press.

Tufte, E. R. 1997. Visual Explanations: Images and Quantities, Evidence and Narrative. Cheshire, Connecticut: Graphics Press.

Ullmer, B. and H. Ishii. 2000. Emerging Frameworks for Tangible User Interfaces. IBM Systems Journal, 9 (3&4) 915-931.

http://tangible.media.mit.edu/content/papers/pdf/ullmer-isj00-tui-framework-pub.pdf.

Van Duyne, D. K., J. A. Landay, and J. I. Hong. 2006. The Design of Sites: Patterns for Creating Winning Web Sites (2nd Edition). Upper Saddle River, New Jersey: Prentice Hall PTR.

Vicente, K. J. 1999. Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Vinh, K. 2006. Getting Real: An interview with Jason Fried. http://www.adobe.com/designcenter/thinktank/gettingreal/. Accessed December 2007.

Von Hippel, E. 2005. Democratizing Innovation. Cambridge, Massachusetts: The MIT Press.

Wenger, E. 1999. Communities of Practice: Learning, Meaning, and Identity. Cambridge, UK: Cambridge University Press.

Waldrop, M. M. 2002. The Dream Machine: J.C.R. Licklider and the Revolution That Made Computing Personal. New York: Penguin (Non-Classics).

Winograd, T., J. Bennett, L. De Young, and B. Hartfield (eds.). 1996. Bringing Design to Software. Reading, Mass: Addison-Wesley Publishing.

Wixon, D., and J. Ramey (eds.). 1996. Field Methods Casebook for Software Design. Hoboken: John Wiley & Sons.

Zuboff, S. 1989. In the Age of the Smart Machine: The Future of Work and Power. New York: Basic Books.

About the Author + FLASHBULB INTERACTION, Inc.

Jacob Burghardt is the founder of FLASHBULB INTERACTION, Inc. He is a research, strategy, and design consultant who specializes in helping product teams to envision powerful, engaging, and productive interactive applications for knowledge work.

Jacob was drawn to a specialty in knowledge work products after seeing how innovative tools in this space can make meaningful differences in the experiences of people practicing their chosen vocations. He views these meaningful differences through the lens of pioneering visions in interactive computing, which optimistically outlined the potential for technologies to augment our cognitive and collaborative abilities as we solve important problems.

In his nine years of experience focused on computing tools for knowledge work, Jacob has worked for industry leading clients on diverse application types, including aviation navigation, scientific instrumentation and data analysis, financial trading, and product lifecycle management.

In his consulting efforts, Jacob seeks to reframe status quo conversations in the development of specialized technologies for knowledge workers. He works with clients to advance their dialog toward a better understanding of design strategy, while emphasizing underlying principles and possibilities in computer mediated work practice.

Jacob challenges clients with new goals and processes, supporting product teams as they envision diverse offerings and sketch elaborative concepts. To inform strategic ideation and ensure real world relevance, he facilitates purposeful, direct, and collaborative conversations between his clients and current or potential end users of their interactive products.

Jacob has lectured and published on topics in user experience. He holds a B.S. in Psychology and a B.S. in Technical Communication from the University of Washington. “Working through Screens” is his first book length publication.

This book was written as the inaugural publication of **FLASHBULB INTERACTION, Inc.**, a consulting firm that works with clients to envision powerful and engaging experiences for knowledge workers at the forefronts of their fields.

We use design thinking to positively transform workplace applications.

We work with clients seeking to set higher goals, ask new questions, and explore innovative answers.

We believe that computing tools for knowledge workers should be envisioned from a perspective that is distinct from those used to conceive consumer products and other technologies.

We help our clients improve peoples’ working lives by advancing tools for thought that target valuable intersections of activity and technological possibility.

We act as trainers, advisors, facilitators — augmenting and inspiring client teams as they envision new and improved onscreen offerings.

We collaboratively investigate, interpret, and visualize application design challenges in a way that drives decision making, shared vision, and meaningful innovation.

We are big believers in concepting relevant futures as a pathway to defining larger strategic directions.

We provide services that target diverse client challenges, ranging from educating product teams, to evaluating and extending existing functionality, to informing application ideation through targeted research, to envisioning entirely new product concepts.

info@FlashbulbInteraction.com
www.FlashbulbInteraction.com

NOTE: Contents of this page exempt from Creative Commons license for this publication.
Copyright 2009, FLASHBULB INTERACTION, Inc.

Working through Screens is the inaugural publication of FLASHBULBINTERACTION, Inc.

This book is available for free in .html and .pdf at www.FlashbulbInteraction.com, where you can also find an abbreviated “Idea Cards” version designed for use in product ideation exercises.

Softcover copies of this book can be purchased at minimum third party cost at:
http://stores.lulu.com/flashbulbinteraction

All original contents of this publication are subject to the Creative Commons license (Attribution-NonCommercial-ShareAlike http://creativecommons.org/licenses/by-nc-sa/3.0/) unless otherwise noted. Please attribute the work to “Jacob Burghardt / FLASHBULB INTERACTION Consultancy.”

Working through Screens is a reference for product teams creating new or iteratively improved applications for thinking work. Written for use during early, formative conversations, it provides teams with a broad range of considerations for setting the overall direction and priorities for their onscreen tools. With hundreds of envisioning questions and fictional examples from clinical research, financial trading, and architecture, this volume can help definers and designers to explore innovative new directions for their products.

"This is gorgeous and insightful." Christina Wodtke

"Beautiful illustrations and useful patterns abound." Jess McMullin

"An excellent, well-illustrated ebook on concept design for designers and product developers." Michael Angeles

"An impressive collection of semi-abstracted design ideas and considerations for knowledge work..." Jonas Löwgren

"I can't wait to delve deeper into this thoughtfully designed document." Keith Tatum

"People will find this book a very useful guide and idea-generator." Judy Ramey