EasyChair Preprint

№ 5853

Team Decision Making in Designing Software for Joint Cognitive Systems

John Flach, Alex Morison, Marisa Bigelow, Jonny Butler and Ariel Swift

June 23, 2021

Team Decision Making in Designing Software for Joint Cognitive Systems

John FLACH, Alex MORISON, Marisa R.G. BIGELOW, Jonny BUTLER and Ariel SWIFT
Mile Two LLC, Dayton, OH, USA

ABSTRACT

The primary function of the software developed at Mile Two is to support human decision making and problem solving. In essence, in designing the software we are designing representations that shape how people experience the decisions and problems that they encounter in their work. The focus of this talk is on how this framing influences our design processes. Like most complex work, we see the design process as an iterative, muddling process that involves generating hypotheses, testing the hypotheses through the creation of artifacts, and then tuning the hypotheses based on the feedback elicited through the artifacts. Further, we realize that this process requires a diverse set of skills and benefits from diverse perspectives. In this talk we will describe our efforts to manage the design team as a joint cognitive system.

KEYWORDS

Software Design & Development; Work Analysis; Representation; Decision Making; Sensemaking.

INTRODUCTION

Mile Two is a software development company that approaches the challenge of design from the theoretical frameworks underlying Cognitive Systems and Resilience Engineering. That is, we consider the primary function of the software to be supporting human decision making and problem solving in joint cognitive systems. In serving this function, the software is a form of representation that plays an important role in shaping how people experience decisions and problems in their work. Thus, in designing software, we are designing a representation of a work domain.

The most important implication of this for the design process is that work analysis to gain a functional understanding of the field of possibilities (e.g., risks and opportunities) within a work domain is essential to good design. You can't design a good representation, if you don't understand the thing being represented. A second implication is that work domains are complex – and that understanding a work domain is at best an iterative process that requires an abductive form of thinking – that requires generating and testing hypotheses. Further, because work domains are complex, there will rarely be a single privileged perspective – understanding a work domain will typically require integration over multiple perspectives. Even further, the design and development processes themselves require a diverse range of skills that span multiple disciplines. Thus, software design typically involves team decision making and problem solving. The goal of this paper is to explore software design and development as a team decision making process.

ITERATIVE DESIGN & DEVELOPMENT

As noted above, design involves an abductive form of reasoning - generating hypotheses, acting on those hypotheses (e.g., creating prototypes/artifacts), and then tuning or revising the hypotheses based on the consequences or feedback elicited through the artifacts. Figure 1 illustrates this as a muddling process. The inset shows an iconic design wheel that breaks down this muddling process into specific subprocesses (e.g., explore, create, observe). And the larger graphic illustrates a path to represent a sequence of activities over time. In the beginning, uncertainty is high as you begin to explore a new work domain, and thus the hypotheses and artifacts generated tend to be low in fidelity. Over time, uncertainty is reduced as you learn about the domain and the fidelity of the hypotheses and artifacts can become increasingly higher in fidelity.

Some important things to note about this process:

- Work Analysis happens continuously throughout the development process. That is, it is not a prerequisite for generating hypotheses or building prototypes. It is a continuous learning process that continues at every stage.
- Hypotheses will be realized as concrete artifacts or prototypes. We are using the term prototype in its most general sense to include any concrete instantiation of an idea (e.g., everything from a sketch on the back of a napkin to a working piece of code or minimally viable product).
- The fidelity of prototypes/artifacts will progressively increase over time. Early in the process a prototype might be a rough sketch of functional relations as a means-ends hierarchy or a concept map. These rough "models" of the work can be refined and further developed into process and state transition diagrams.

More concrete prototypes can then be generated as wireframe sketches, which in turn can become interactive prototypes, then MVPs and eventually operational quality software.

- The prototypes/artifacts generated in early stages become important seeds for the work analysis in later stages. As Shrage (2000) observed in "Serious Play" – having a concrete artifact to interact with can be essential to the learning and discovery process. It is often difficult for people to describe how they do their work or to identify what they need from software in the abstract. However, when interacting with a concrete artifact they will often recognize problems and/or opportunities that they would not have thought of otherwise. In essence, generating concrete prototypes is essential to the perception – action dynamic that is critical to naturalistic decision making. Also, sometimes low fidelity prototypes can be particularly useful in eliciting feedback, since sometimes people will be reluctant to suggest changes to an artifact that looks too polished or final.
- Thus, prototypes are important for engaging stakeholders in a participatory design process. And because the development process can only approach complete knowledge of a domain asymptotically, engaging the domain stakeholders can be critical for determining when a design is "good enough."

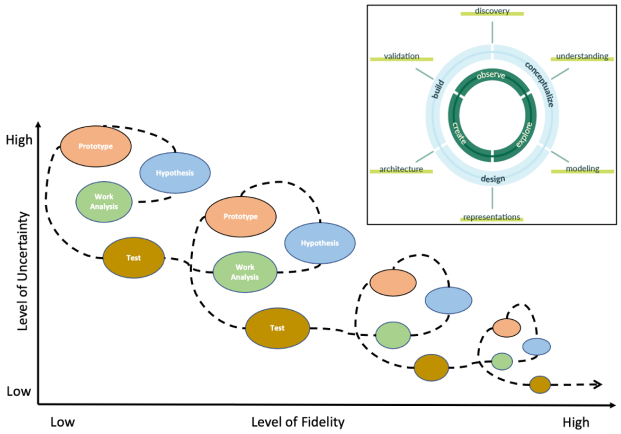

Figure 1. The design and development trajectory involves an iterative learning process in which uncertainty is progressively reduced and fidelity is progressively increased.

WORK ANALYSIS

The goal of work analysis is to understand the work domain in terms of the values, goals, and risks (the why) and the actions, decisions, and problems that arise in accomplishing the functional objectives (the how). Figure 2 illustrates some of the questions that need to be addressed through work analysis in relation to the Abstraction-Decomposition space described by Vicente (1999). In contrast to some others, we treat this framework as a guide to the territory that needs to be addressed in work analysis, but not as an artifact for representing what is learned during work analysis. Rather, we use a wide range of artifacts as both probes for exploring the domain and as formats for representing what we learn. Note that many of the artifacts can be used productively in different regions of this space. For example, scenarios that represent critical incidents can be useful for probing the value constraints and how people deal with goal conflicts and trade-offs, and they can also be useful for understanding the flow of activities in terms of process constraints. The key distinction is that unlike many advocating for a CSE approach to design, we don't treat the Abstraction Hierarchy as a single artifact, but rather as a collection of artifacts that can be used to guide an exploration process that spans the Abstraction-Decomposition space in a productive way.

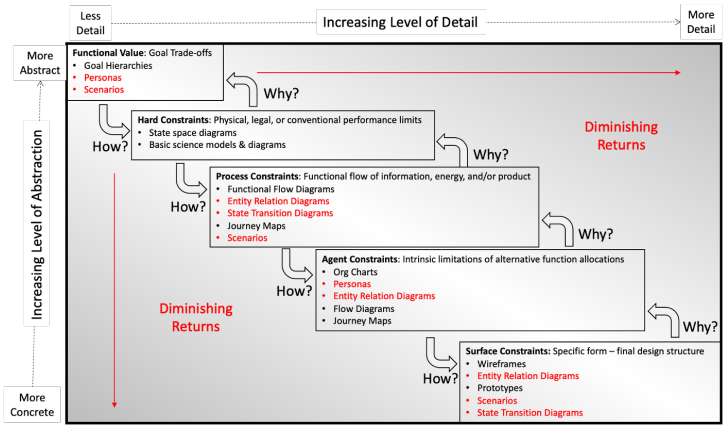

Figure 2. The Abstraction-Decomposition Space provides a framework for guiding exploration of a work domain to understand why and how people work. However, a collection of artifacts are needed as probes and representations for describing what is learned (Flach, Bennett, Butler & Heroux, In press).

As illustrated in Figure 1, work analysis is not a prerequisite but a co-requisite of the idea generation, prototype development, and testing process. That is, it is ongoing throughout the development process and prototypes/artifacts generated at earlier stages of the process become essential probes for knowledge elicitation and learning during later stages of the process.

TEAM OF DISCIPLINES

The choice of the word 'team' is intended to emphasize that producing software requires more than a collection of disciplines – it requires 'teamwork' or 'team cognition.' Challenges include creating common ground to facilitate communications and coordination within an organization that includes people with different backgrounds and diverse skill sets. And it requires attention to the social dynamics of group interactions to facilitate group problem solving. For example, it requires leadership to moderate communications to ensure that diverse perspectives can be voiced and be listened to and appreciated (e.g., Pentland, 2015).

Figure 3 shows some of the disciplines and roles that make up the design teams at Mile Two. The larger graphic shows four 'design' disciplines that play an important role in shaping our goals for the user experience that our software will help engender. In the academic world – these different disciplines are often debating or competing to prescribe the skills and ideals for "good" design. However, the graphic is intended to reflect the fact that all of these disciplines contribute unique insights into the user experience, and that none of these disciplines spans the full experience. Thus – collaboration is essential.

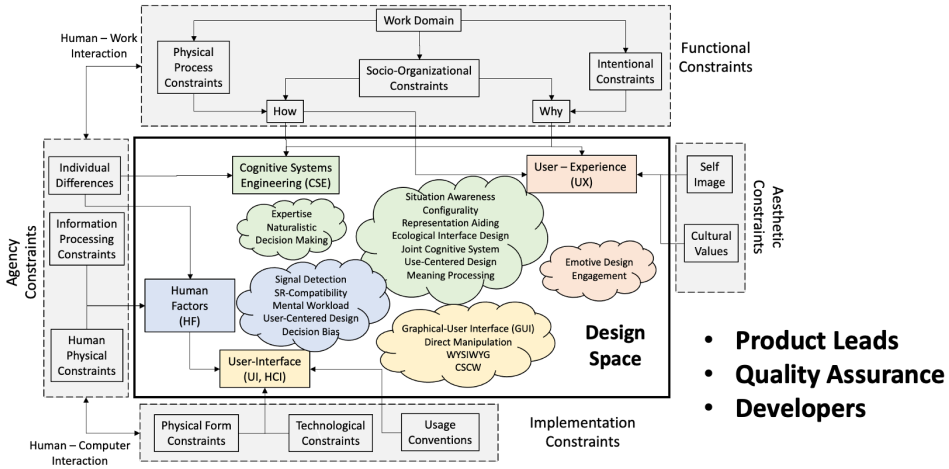

Figure 3. The graphic illustrates that multiple disciplines bring important perspectives to modelling the user experience. However, in addition to these disciplines other roles are needed – Product Leads, Quality Assurance, and Developer are essential roles on our design teams. (Flach, Bennett, Butler & Heroux, In Press).

In addition to the 'design' disciplines, there are several other important roles on Mile Two design teams. *Project Leads* are primarily responsible for making sure the trains run on time and within budget. They tend to be the primary liaison with the 'customers' who fund our work. Further, they tend to take responsibility for 'coordination' and for managing the social dynamics within the team. The *Quality Assurance (QA)* role takes responsibility for all aspects of quality – this includes consistency in the user experience and functionality of the code (does it do what it is intended to do? Are there bugs in the code?). The QA role brings an attention to detail that is above and beyond what is typically associated with usability testing. And finally, there is the role of developer. The developers design the architecture for the code (e.g., interface to APIs, the data structures, and functional flow among the components) and do the actual software coding. Typically, teams include a senior developer who designs the architecture and manages the coding process and several programmers who actually write the code.

In studying teamwork, one of the challenges for team coordination is 'handoffs' typically associated with shift changes in domains such as process control and healthcare. On one hand, there is always a danger that essential aspects of the context will not be communicated during the handoff and that the incoming shift will not have full situation awareness for dealing with challenges during their shift. On the other hand, sometimes a shift change can bring a new perspective to a problem that helps the current shift escape from an unproductive framing of a difficult problem. In design, handoffs are more likely to be associated with passing information across disciplinary silos – e.g., one group does the work analysis and then hands off the results to others who create the interface representations, who hands off their results to another group who does the actual coding. At Mile Two we have developed our design process to minimize the risks of losing

context, while at the same time amplifying the benefits of multiple perspectives. To do this, we involve the entire team at every phase of the design process. This means that project leads, quality assurance, and developers all participate in the work analysis and hypothesis generation stages of development. This helps to ensure that when it comes time for coding and quality testing the people doing this work share an understanding of the work domain in terms of the functions that the software is intended to support and the nature of the decision making and problem-solving activities involved in accomplishing those functions. In this respect, the products of work domain analysis become the common ground for communications within the design team.

It is important to acknowledge that this is not necessarily easy to realize, and that it can be quite a stretch for people who are used to working within their disciplinary silos. Thus, Mile Two has made a significant investment to introduce and train all our staff to appreciate the theoretical motivations behind work analysis (i.e., principles of cognitive systems and resilience engineering). This has included extended workshops taught by Dave Woods and an ongoing CSE working group, where we introduce some of the artifacts shown in Figure 2 and give people guided practice in using them.

Finally, while all the disciplines and roles participate in every phase of the design and development process, certain roles and skills will take the lead at different phases of the design process. For example, CSE and UX designers will play an important role during the early stages of work analysis leading the exploration of work constraints and conducting knowledge elicitation sessions; UX and UI designers will play a leading role in translating the understanding of the work that emerges into interface representations; and HF and QA will tend to lead during the final stages to test usability and to ensure that the final software is free of bugs. Thus, our design teams function as heterarchies in that authority shifts between different disciplines as a function of the match between their skills and the demands of different phases of development. However, the full team contributes at every phase of the design and development process, and the leads of a particular phase are not always the sources of the most important insights.

ARE WE THERE YET?

One of the major challenges in any creative problem solving activity is to decide when to let go of a problem. At what point is the solution (artifact/prototype) 'good enough' to move onto the next phase, and ultimately when is the final product ready to be released for operational use. At best, understanding a work domain is an asymptotic process due to the complexity of most work domains; and that assumes that the work domain is a stationary target. However, in reality nearly all work domains are moving targets that are continuously evolving in response to new opportunities (e.g., technological innovations) and new challenges (e.g., dynamic markets and/or competitors). Although all responsibilities are shared across the team – the Project Lead has the ultimate authority for deciding what is 'good enough' for the current phase or design sprint, and when to move onto the next phase. In making this call the Product Lead has to consider the cost and time constraints imposed by the ultimate customer for the product. Thus, to a large extent, the ultimate authority for what is 'good enough' rests in the hands of the customer who is paying for the software and/or the operators who will ultimately be using the software.

One of the functions of the prototypes/artifacts that are generated in the different phases of design is to provide a means for engaging customers/operators as participants in the design process. Again, in line with the spirit of "Serious Play" the artifacts provide a means to get feedback from the domain stakeholders to test assumptions and hypotheses about the work domain. Ideally, the artifacts allow the domain stakeholders to communicate their prior experiences and also to gain new, and deeper insights into their work domain and into the possibilities for innovation that new software tools might offer. The reaction of the stakeholders to the artifacts provides important feedback for determining when an artifact is 'good enough' and when it is time to move on to the next phase of development. However, it is not a strictly linear process, and often, discoveries at later phases suggest ways to revise and change artifacts generated in earlier stages.

SUMMARY & CONCLUSION

To sum up, Mile Two is attempting to apply the lessons from the last 30 years of research inspired by the Cognitive Systems and Resilience Engineering frameworks to our own work in developing software to support cognitive work. Thus, we consider design teams to be joint cognitive systems where multiple people, with a variety of backgrounds and skills come together to solve complex problems. We have framed the design problem as designing representations to support cognitive work. In this effort, we look to work analysis as a means to create a common ground for understanding the problem and we do our best to anticipate and address the challenges associated with team cognition and organizational sensemaking.

In describing our process, it is difficult to avoid creating the impression that we have a well thought out, detailed road map or recipe that other organizations might easily adopt. When in reality we are still very much in a muddling phase of development with fairly strong hypotheses about how things should work – but still in a novice stage of implementing those hypotheses. In part, we believe that because of the complex, creative nature of design work itself and the challenges of getting a diverse collection of people and personalities to trust and listen to each other – maybe mindful muddling is the best that can be achieved. At the end of the day, the ultimate test of our mindful muddling will be the products that we create.

ACKNOWLEDGEMENTS

We want to thank the Mile Two management for supporting us through the muddling process and for providing training to help us succeed in this work. We also want to thank the many clients who have invested in our work and who have participated in the design process.

REFERENCES

Flach, J.M., Bennett, K.B., Butler, J.W. & Heroux, M.A. (In press) Representation Design. In G. Salvendy & W. Karwoski. *The Handbook of Human Factors and Ergonomics*. Hoboken, NJ: John Wiley & Sons.

Pentland, A. (2015). *Social Physics: How social networks can make us smarter.* New York: Penguin Books.

Shrage, M. (2000). *Serious Play: How the world's best companies simulate to innovate.* Boston, MA: Harvard Business School Press.

Vicente, K.J. (1999). *Cognitive Work Analysis.* Mahwah NJ: Erlbaum.