

Position Paper: Designing for Usefulness for Complex Problem Solving

Barbara Mirel, University of Michigan, School of Information

This position paper focuses on improving the science of design as it relates to applications for complex problem solving. A good deal of research and development target technologies for complex problem solving, such as multi-agent systems for distributed problem solving, data mining technologies, and interactive data visualizations for exploratory analysis. Unfortunately, these advances have not converged into truly useful applications for end-users' real work in context. The science of design in research and development into complex problem solving is not a science of designing for *usefulness*. This paper explains the needs for and issues involved in designing for usefulness for dynamic and emergent problem solving in context.

Designing for usefulness requires unifying research and practice. Usefulness by definition is grounded in practice. To be useful, software must accommodate the distinguishing dynamics of complex problem solving, all shaped by context and practice. They are: Ill-defined situations; vague or broad goals; large volumes of data from many sources unprocessed for immediate purposes; nonlinear, often uncharted analytical paths; no pre-set entry or stopping points; many contending legitimate options; collaborators with different priorities; "good enough" solutions with no one right answer; and underlying patterns structuring open-ended investigations that, due to contextual conditions and constraints, are never performed the same way twice.

I maintain that complex problem solving is a distinct breed of design, different from designing for well-structured work. Complex problem solving requires its own design thinking, methodologies, and choices. Unfortunately, researchers and developers do not typically approach it as such, and end up producing good designs but for the wrong problems. Design groups, therefore, must re-structure development cycles to integrate software and usability engineering from the start of a project on. Usefulness via upfront user and task analyses and user modeling must influence decisions about product scope, architecture, and functionality and not just user interfaces.

Once integrated software and usability engineering is in place, designing for real world complexity requires new assumptions in design thinking, and its methodologies, strategic emphases, and choices draw on but modify common user-centered design approaches. Designing for usefulness addresses issues in the call for position papers and implies others, as follows:

- **The role of problem formulation and purpose of design:** Designing for users' dynamic and emergent problem solving requires close attention to defining the right unit of analysis. Few design issues are more important, more challenging, and more neglected than getting the level of detail right in the unit of analysis. In designing for complex problem solving, a unit of analysis is too detailed if set on logical tasks, task repertoires, individual users or groups of users; it is too broad as an activity system. Defining the problem situation as the unit of analysis strikes the right balance and highlights salient issues for analysis and modeling.
- **Assumptions about support for users' work.** Five assumptions about support for users' work often drive designs for software and software intensive systems but are counter-productive for complex problem solving. These assumptions and more appropriate alternatives or emphases follow:

| Old Assumptions | New Design Thinking |
|--|--|
| The whole of work equals the sum of its parts. | Complex work is synergistic. |
| Users need to assimilate the program model and logic | Programs have to adapt more to users. |
| Generic support is best for reusability. | Domain-specificity is best for user-adaptability. |
| The goal of support is to simplify work. | The goal is to be operationally simple but intellectually sophisticated and nuanced. |
| Problem solvers are rational agents. | They are extemporaneous actors in emerging plots. |

- **Methodologies for user experience studies:** Because designers cannot possibly presuppose all of users’ moves, strategies, and knowledge for complex problem solving, methods in user and task analysis must focus on capturing underlying patterns of inquiry in types of problems for specific domains and the variability tied to interactive conditions, contingencies, serendipity, and other idiosyncrasies. To uncover these patterns of inquiry, designers must conduct field observations -- a non-negotiable method in designing for usefulness.
- **Representations and notations for designs:** Typically, user experience findings are represented as scenarios, use cases, user profiles, consolidated diagrams, or UML notations. These forms, however, often let complexity and contextual dynamics fall through the cracks. By contrast, representing patterns of inquiry as task landscapes and as movements through interactive conditions and constraints fosters sustained attention to regularity *and* variability and to contextually-driven wayfinding and sensemaking activities. These forms guard against premature leaps to individual tasks and objects. Aptly representing models of complex work is crucial. They are the basis for deciding on some of the hardest yet most important choices (see next bullet) in designing for complex problem solving.
- **Conceptual design strategies and choices:** As with the previous assumptions, many strategies and choices that may be effective for well-structured work are inappropriate for open-ended, multi-pronged inquiries. For the latter, designers need to draw on some of the most relevant current advances and inventively synthesize them into elegant designs -- offering integrated support for integrated albeit open-ended investigations. This “toolkit” for creating integrated designs includes current work on domain-specific architectures, multi-agent technologies, problem-centered data visualization appliances, and spatially-rather than procedurally-oriented support for discovering the structure of a problem and solution. Here it becomes evident that more research is needed in regard to some of the thorniest issues and trade-offs in designing for usefulness. These issues include deciding how to distribute control between users and software (what to automate and what not to); how to design effective rich data displays for specific pattern-based needs; when to provide rich vs. serial data displays; how to support *cumulative* querying, analysis, and comparisons; and how to structure workspaces for integrating and analyzing heterogeneous data from many sources.
- **Evaluation of software and software-intensive system designs:** Design processes for complex problem solving must extensively test the user model. Since its appropriateness depends on software’s fit within the whole system of work, this evaluation needs to take place in the field, ideally with users defining their own real work problems and paths through them. This iterative testing for usefulness must precede any usability testing.

About the author: My book, *Interaction Design for Complex Problem Solving: Developing Useful and Usable Software* (Elsevier./Morgan Kaufmann, 2003) is the first that I know of that explains and exemplifies designing for complex problem solving as a distinct breed of design in a way that unifies theory and practice. My research in this area spans 10 years as a tenured professor; three years as a senior human factors specialist in industry, leading usability efforts in software for complex problem solving, including data visualizations and a patent for a visual discovery design; and now as a visiting associate professor and research investigator in the School of Information at the University of Michigan. I am a co-PI on an NSF ITR grant (“Collaborative Augmentation of Knowledge Production”). Two of my other current projects related to software-intensive systems involve designing for lay surveillance staff to help them detect fraud in Medicaid and Medicare healthcare record systems and creating tools for advanced troubleshooters who collaboratively analyze confounding problems in hybrid, voice-over-IP networks. In addition, I teach graduate courses in Information Visualization at the UM and have published extensively on support for complex tasks, data visualizations, usability testing results, and methodologies for user and task analysis.