Scenarios and Design Cognition

John M. Carroll
Center for Human-Computer Interaction and Department of Computer Science
Virginia Tech, Blacksburg, VA 24060-0106 USA
carroll@cs.vt.edu

Abstract

Scenarios of human activity are widely used in many facets of the design of software systems. This methodological development can be understood in terms of design cognition: Scenarios support and enhance the solution-first strategy, which typical of, and perhaps fundamental to expert problem-solving for "ill-defined" problems like design and planning.

1. Introduction

Scenario-based design is a family of techniques in which the use of a future system is concretely described at an early point in the development process. Narrative descriptions of envisioned usage episodes are then employed in a variety of ways throughout the development of the system. Scenario-based design changes the focus of design work from defining system operations to describing how people will use a system to accomplish work tasks and other activities.

Scenario-based practices now span most of the software development lifecycle: Requirements development and analysis, envisionment and design development, user interface design and prototyping, software development and implementation, documentation and training development, formative and summative evaluation.

The pervasive and growing use of scenarios in the development of systems and software can be understood in terms of design cognition: Scenarios support and enhance the solution-first strategy [6], which is typical of expert performance ill-structured problem contexts such as planning and design. Scenarios evoke reflection and inquiry in design, facilitate suspension of commitment, promote work-orientation, and encourage multifaceted brainstorming and development.

This analysis of design cognition helps to set a research agenda for better understanding circumstances under which scenarios may be more and less effective in design, contrasts among various different types of scenario-like objects, organizational accommodations that are required for adopting, developing and supporting scenario-based design practices, and the relationship of scenario-based methods to new and established design concepts and techniques.

2. The Solution-first Strategy

The solution-first strategy involves generating a provisional design solution as a method for identifying requirements. In other words, one solves the problem in order to define the problem.

This seems paradoxical, until one considers some of the defining properties of design considered as problem-solving. Design is an example of what Reitman [8] called *ill-structured* problems (similar conceptions were articulated subsequently by Rittel, Simon, and others). Reitman defines ill-structured problems as those for which (1) the problem state is not fully specified, (2) possible moves in the problem space are not all given, and (3) the goal state is not specified. In other words, designers, in general, do not know the current situation, what they can do to transform it, or what target situation they are trying to enable.

Reitman's characterization is not about deficiencies in current practices. It is about the essence of design. Designers cannot have full descriptions of the current problem state, because in principle that would be a full description of the world. They cannot have a list of allowable transformations, because that would presuppose a comprehensive description of the world, and of all worlds reachable from this world. And they cannot have an adequate specification of the goal state, because having that would answer the design problem, dissolving it into a construction problem.

Empirical studies of design describe how, and to some extent why designers employ the solution-first strategy. For example, our research group studied a

Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE'02)
1090-705X/02 $17.00 © 2002 IEEE

systems engineer and a librarian designing distributed access to reference materials [5], [7]. Their design activity produced a set of subsolutions, each addressing a cluster of requirements. At the end of this process, they abandoned all the subsolutions, prioritized and selected requirements, and produced a final solution that was not related to any of the subsolutions.

The subsolutions helped the designers to reason concretely about their requirements, without the commitment of a single-thread waterfall. They provoked analogies and connections, and raised new questions and constraints, expanding the design space. They served as a vehicle for better understanding the problem state and its possible transformations, through sketching the partial goal states entailed.

One of the standard failure patterns for junior designers is to become fixated in the details of problem definition, and to never emerge with a candidate solution [6]. Apparently, one of key lessons that designers must learn is to create and work with early solutions [2], [10].

3. Solving ill-structured problems

The solution-first strategy is neither a logical necessity nor a methodological panacea. It has well-known and serious risks. One risk is that designers who construe their activity as directed at early solutions, may generate solutions before sufficiently analyzing what is already known about the problem and the possible moves. This perverts the logical indeterminacy of design (namely, the fact that we cannot have a full description of the world) into a kind of know-nothingism (to wit, why know anything if you can't know everything). This would clearly result in less valuable partial solutions, or even in partial solutions that obscure key pieces of relevant information. Another risk is that adopting a single solution hypothesis as a means of generating further requirements, solution refinements, and new candidate solutions can cause designers to fixate on a suboptimal or incorrect solution approach. In this way, an attractive candidate solution can become a cognitive obstacle to enumerating an adequate space of alternative solutions.

Designers may too readily reuse a solution approach they have employed before, one that is familiar and accessible, but perhaps not appropriate in the current circumstances. They may generate an insufficient variety of partial solutions and thereby insufficiently explore the problem space. They may insufficiently analyze their own partial solutions; discovering some, but not all of the critical requirements. They may not be able to integrate partial solutions. Or they may have trouble abandoning an initial solution approach that has served its purpose [6].

Given (a) that the solution-first strategy is both motivated by fundamental characteristics of design cognition, and empirically attested in the design of software systems, as well as in other design domains, and (b) that the strategy is associated with a variety of methodological problems, it is important to consider how the solution-first strategy can be more effectively supported in design practices. One could define and standardize processes to ensure that the primary risks are better controlled (for example, requirements gathering and analysis as distinct and mandatory upstream activities). One could develop tools to ensure that primary benefits of the solution-first strategy are reliably obtained. One could try to do both [4].

4. Scenarios as a tool in solution-first design

Envisioning and analyzing scenarios of human activity supports an implementation of the "solution-first" strategy to design while controlling some of the main risks. Scenarios emphasize reflection and inquiry as design activities, encourage suspension of commitment, promote work-orientation, and help to integrate diverse design activities.

Working with scenario descriptions evokes reflection in the context of solution-first design activity. They evoke empathy for the participants in an envisioned situation of use, raise questions for designers to address as they elaborate the design, and emphasize the dynamic flow of activity in the use of a designed artifact or system.

Scenarios are concrete and flexible. Their form alone suggests they are tentative and incomplete, but they are also vivid. This makes them evocative, but also helps differentiate them from "real" solutions. Since scenarios are merely envisionments, they can't be confused with implemented solutions, and are easily revised or abandoned. This helps designers manage the uncertainty of design situations — both by helping designers to avoid premature closure in their designs and by evoking new ideas to elaborate designs.

Scenarios anchor design discussion in the work to be supported, encouraging input and participation among all stakeholders. They serve as "advance organizers" to functional specifications [1].

Scenarios can be written at multiple levels, from many perspectives, and for many purposes. They afford multiple views of an interaction, diverse kinds and amounts of detailing, helping designers manage the many consequences entailed by any given design move. They encourage analysis and guide the integration of partial solutions.

5. Scenario processes in solution-first design

Scenarios, like most representational tools bring few guarantees. Today, it is sadly common to hear designers complain that they wrote a scenario, but their design still failed in all the usual ways. The key to this riddle is that

designers *make use* of scenarios, not just write them [3], [9]. For example, scenarios trigger their own evaluation by being obviously and inherently incomplete descriptions of situations in the world. Yet designers still must react to this incompleteness by articulating and pursuing the questions raised by their scenarios. In other words, the designer must ensure that each scenario is a requirements generator and not just a future system envisionment.

Similarly, scenarios can help designers avoid premature commitment, by being malleable and sketch-like. They can remind designers by their form alone that the set of possible design outcomes is open-ended, and that hypothetical reasoning is important and appropriate. But if designers choose to treat scenarios merely as vague specifications, instead of as something entirely different from specifications, they will chiefly realize the risks of the solution-first strategy, and not the benefits.

By vividly representing work activity, scenarios can help designers stay focused on the end-user's activity and experience, and avoid confusing their own preferences with those of the users. But if designers write blue-sky descriptions of mythical users, if they never visit workplaces, if they do not include users on the design team then the scenarios they develop may only codify their misunderstandings and misdirect their design efforts.

And finally, although scenarios are multifaceted and can be interpreted and used in a great variety of ways, designers have to organize and carry out that use. They must switch modes from depth to breadth, from user activity to software use cases, from implementation options to contrasting design rationales. Writing a scenario is necessary but far from sufficient to achieve an effective scenario-based implementation of the solution-first strategy.

6. Conclusion

The solution-first strategy is more than a current technical challenge. It is an underlying dynamic in design as human problem-solving activity. Understanding the solution-first strategy can help us to understand design better, and perhaps to better support designers.

Solution-first design entrains characteristic hazards, [6]. Designers tend to generate solutions too quickly, before they analyze what is already known about the problem and possible moves. Once an approach is envisioned, they may have trouble abandoning it when it is no longer appropriate. Designers may too readily reuse pieces of a solution they have used earlier, one that is familiar and accessible, but perhaps not appropriate. They may not analyze their own solutions very well, or they may consider too few alternatives when exploring the problem space.

These hazards may be ameliorated when scenarios of use are the medium for early design solutions, and are employed as tools for generating, investigating, and integrating problem analyses and partial solutions.

References

[1] Ausubel, D.R. 1960. The use of advance organizers in the learning and retention of meaningful verbal material. *Journal of Educational Psychology, 51,* 267-272.

[2] Beck K. 1999 *Extreme Programming Explained: Embrace Change.* Reading MA: Addison-Wesley.

[3] Carroll JM, (2000) *Making use: Scenario-based design of human-computer interactions,* MIT Press, Cambridge, Massachusetts.

[4] Carroll, J.M. & Rosson, M.B. 1987. The paradox of the active user. In J.M. Carroll (Ed.), *Interfacing thought: Cognitive aspects of human-computer interaction.* Cambridge: MIT Press/Bradford Books, pp. 80-111.

[5] Carroll, J.M., J.C. Thomas, & A. Malhotra. 1979. A clinical-experimental analysis of design problem solving. *Design Studies, 1,* 84-92.

[6] Cross, N. 2001. "Design cognition: Results from protocol and other empirical studies of design activity." In C. Eastman, M. McCracken & W. Newstetter (eds.), *Design knowing and Learning: Cognition in Design Education*. Amsterdam: Elsevier, pages 79-103.

[7] Malhotra, A., J.C. Thomas, J.M. Carroll, & L.A. Miller. 1980. Cognitive processes in design. *International Journal of Man-Machine Studies, 12,* 119-140.

[8] Reitman, W. R. 1965. *Cognition and thought.: An information processing approach.* New York: John Wiley and Sons.

[9] Rosson, MB and Carroll JM 2002. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction.* San Francisco: Morgan Kaufmann.

[10] Wasserman, A.I. & Shewmake, D.T. 1982. Rapid prototyping of interactive information Systems, *ACM Software Engineering Notes*, 7(5), 171-180.