7 Scenario-Based Design

John M. Carroll and Steven R. Haynes

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 to guide the development of the system. Scenario-based design changes the focus of design work from defining system operations (that is, functional specification) to describing how people will use a system to accomplish work tasks and other activities.

Scenario-based design elaborates a traditional theme in human factors and ergonomics, namely, the principle that human characteristics and needs should be pivotal considerations in the design of tools and artifacts. In scenario-based design, descriptions of usage situations become more than orienting examples and background data, they become first-class design objects. Scenario-based design takes literally the adage that a tool is what people can do with it, and the consequences for them and for their activities of using it.

ORIGINS: STRATEGIC PLANNING AND STRATEGIC MANAGEMENT

During the early years of the Cold War, strategic planners came to rely on scenarios to anticipate and analyze future contingencies. The situations they were concerned with were complex and unfamiliar. Scenarios facilitated vivid and broad-scope analyses that encouraged consideration of contextual details and temporal dynamics that were sometimes overlooked in more formal contingency planning. The simple rubric of time facilitated comprehension and integration of the many interacting elements in a problem situation, while emphasizing to analysts that circumstances are constantly changing. Scenarios were found to stimulate imagination, helping analysts generate new contingencies and relationships. Finally, scenarios reduced the perceived costs of considering more alternatives. The "accidental war scenario," in which a mistaken launch of a single tactical missile somewhere in Europe triggers a cascade of events, culminating in all-out global war, was discovered through scenario analysis and became a touchstone of strategic planning for a generation. Prior to the late 1940s, no planner had ever considered such escalating contingencies (or cataclysmic outcomes).

Beginning in the 1970s, scenario-based methods were widely adopted and developed in corporate strategic management. Scenario analysis was credited with Royal Dutch Shell's success during the mid-1970s oil crisis. Since 1980, scenario-based techniques have developed in key areas of computer system and software design—requirements engineering, human–computer interaction, and software design. These separate lines of methodological development have begun to converge around the proposal that various scenario-based practices can be mutually leveraged throughout the system development process.

WHAT IS SCENARIO-BASED DESIGN?

In scenario-based design, key situations of use in established work practice are enumerated and described in schematic narratives. Each narrative depicts one or more actors, a defining goal and possibly several ancillary goals, a context of tools and other artifacts, and a sequence of actions and events through which the defining goal is achieved, transformed, obstructed, and/or abandoned. Figure 7.1 describes the use of a handheld, portable maintenance device (PMD) for Marine Corps light armored vehicle mechanics. At an early phase of design, this might be an envisioned scenario, vividly conveying a new system concept, or, if this type of system is already in use and being refined, this might be an observed scenario, summarizing typical use and current design challenges (for example, how to get users to carry out practice exercises). In either case, the scenario guides the development of requirements and prototypes, and the specification of system functions. Subsequently, it could guide the development of training and documentation, and the design of evaluation tasks.

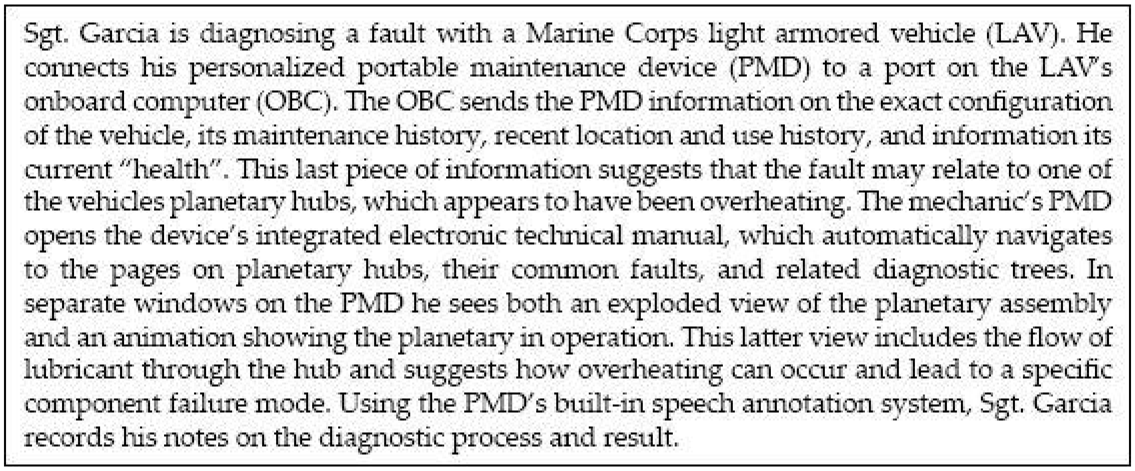

Sgt. Garcia is diagnosing a fault with a Marine Corps light armored vehicle (LAV). He connects his personalized portable maintenance device (PMD) to a port on the LAV's onboard computer (OBC). The OBC sends the PMD information on the exact configuration of the vehicle, its maintenance history, recent location and use history, and information its current "health". This last piece of information suggests that the fault may relate to one of the vehicles planetary hubs, which appears to have been overheating. The mechanic's PMD opens the device's integrated electronic technical manual, which automatically navigates to the pages on planetary hubs, their common faults, and related diagnostic trees. In separate windows on the PMD he sees both an exploded view of the planetary assembly and an animation showing the planetary in operation. This latter view includes the flow of lubricant through the hub and suggests how overheating can occur and lead to a specific component failure mode. Using the PMD's built-in speech annotation system, Sgt. Garcia records his notes on the diagnostic process and result.

Figure 7.1 A scenario of use for a Marine Corps portable maintenance device (PMD)

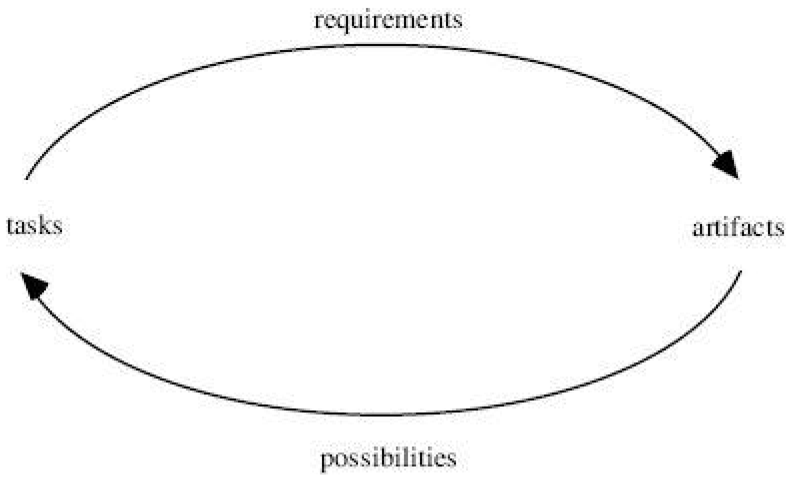


Figure 7.2 The task-artifact cycle

Note: Tasks help articulate requirements for new technology that can support them; designed artifacts create possibilities (and limitations) that redefine tasks.

The ability of scenarios to vividly depict both current and future situations has encouraged their use as working representations in a wide variety of disciplines, from strategic management and operations research to requirements engineering, object-oriented design, and human–computer interaction. In the large, work and technology co-evolve: people engage in tasks and activities. They make discoveries and encounter difficulties; they experience insight and satisfaction, frustration and failure. At length, their tasks and experiences help to define requirements for future technology. Subsequent technological tools, in turn, open up new possibilities for tasks and activities, new ways to do familiar things, and entirely new things to do. They entrain new skills to learn and perform, new opportunities for insight and satisfaction, and new pitfalls of frustration and failure. Ultimately, the new tasks and new technology become a new baseline, helping to evoke requirements for further technology development. Scenarios have been proposed as an integrative representation of work and technology for managing this pattern.

HOW DOES SCENARIO-BASED DESIGN WORK?

Scenario-based design addresses five key technical challenges in the design of technology: Scenarios evoke reflection in the content of design work, helping developers coordinate design action and reflection. Scenarios are at once concrete and flexible, helping developers manage the fluidity of design situations. Scenarios promote work-oriented communication among stakeholders, helping to make design activities more accessible to the great variety of expertise that contributes to design, and addressing the challenge that external constraints designers and clients often distract attention from the needs and concerns of the people who will use the technology. Scenarios afford multiple views of an interaction, diverse kinds and amounts of detailing, helping developers manage the many consequences entailed by any given design move. Finally, scenarios can be abstracted and categorized, helping designers to recognize, capture, and reuse generalizations, and to address the challenge that technical knowledge often lags the needs of technical design.

Design Action can Undermine Reflection: Scenarios Evoke Reflection in Design

Constructing scenarios of use as focal design objects inescapably evokes reflection in the context of doing design. The scenario in Figure 7.1 succinctly and concretely conveys a vision of student-directed, multimedia instruction. It is a coherent and concrete vision, not an abstract goal, not a list of features and functions. Elements of the envisioned system appear in the scenario embedded in the interactions that will make them meaningful to people using the system to achieve real goals—perhaps revelatory, perhaps cryptic, but definitely more than just technological capabilities. For example, the role of natural language query is exemplified as a means of locating further case studies that illustrate the principles of harmonic motion.

The scenario emphasizes and explores goals that a person might adopt and pursue, such as watching the film clips twice, or skipping the exercises. Some of these goals are opportunistic, such as investigating the Tacoma Narrows collapse because of experience with a totally unrelated bridge collapse, or deciding to branch from bridge failures to flutes. The scenario implicitly articulates the usage situation from multiple perspectives: the student stores and annotates a video clip with speech, raising specific requirements for user interface tools and presentation as well as for particular data structures and memory. The scenario impels the designer to integrate the consideration of such system requirements with consideration of the motivational and cognitive issues in education that underlie the person's actions and experiences.

Design Problems are Complex and Fluid: Scenarios are Concrete but Flexible

Scenarios reconcile concreteness and flexibility. They are concrete in the sense that they simultaneously fix an interpretation of the design situation and offer a specific solution: the scenario in Figure 7.1 specifies a particular usage experience that could be prototyped and tested. At the same time, scenarios are flexible in the sense that they are deliberately incomplete and easily revised or elaborated: in a few minutes, a piece of the scenario could be re-written (for example, stipulating a system-initiated prompt for the associated course module on harmonic motion) or extended (for example, the objects in the scenario could be described as anchors for a variety of link types).

Scenarios embody concrete design actions and evoke concrete move testing on the part of designers. They allow designers to try things out and get directive feedback. But their flexibility facilitates innovative and open-ended exploration of design requirements and possibilities, helping designers to avoid premature commitment. This is especially critical in that design problems are ill defined: they can always be understood and approached in multiple ways. Interactively exploring design problems often helps to reveal more about them.

The power of scenarios to convey concreteness through tentative and sketchy descriptions derives from the way people create and understand stories. Like the strokes of an expressionist painting, scenarios evoke much more than they literally present. The human mind seems especially adept at overloading meaning in narrative structures, both in generation and interpretation, as illustrated by the remarkable examples of dreams and myths. Indeed, dreams and myths are universal tools for coping with uncertainty. Part of their power stems from being recalled and discussed throughout a community. In a more modest scale, design teams do this too: Sharing and developing scenarios helps to control the uncertainties inherent in the work, while strengthening the team itself.

Design is Constrained by External Factors: Scenarios Promote Work Orientation

Scenarios are work-oriented design objects. They describe systems in terms of the work that people will try to do as they make use of those systems. A design process in which scenarios are employed as a focal representation will ipso facto remain focused on the needs and concerns of prospective users. Thus, designers and clients are less likely to be captured by inappropriate gadgets and gismos, or to settle for routine technological solutions, when their discussions and visions are couched in the language of scenarios, that is, less likely than they might be if their design work is couched in the language of functional specifications. Design work is often constrained by various external factors in the development organization and the marketplace (policies, standards, competitive products, past and planned products, schedules, resource budgets). Scenarios help keep attention focused on the work activity to be supported.

Scenarios help to integrate the variety of skills and experience required in a design project by making it easier for different kinds of experts to communicate and collaborate. They support the direct participation of users and other client stakeholders, helping to anticipate organizational impacts of new technology. Scenarios also ease problems associated with handoffs and coordination in the design process by providing all groups with a guiding vision of the project goal to unify and contextualize the documents and partial results that are passed back and forth.

Scenarios directly support the development of summative evaluation tasks. But perhaps more importantly, maintaining work-oriented scenarios throughout the development process allows formative evaluation walkthroughs to be carried out continuingly.

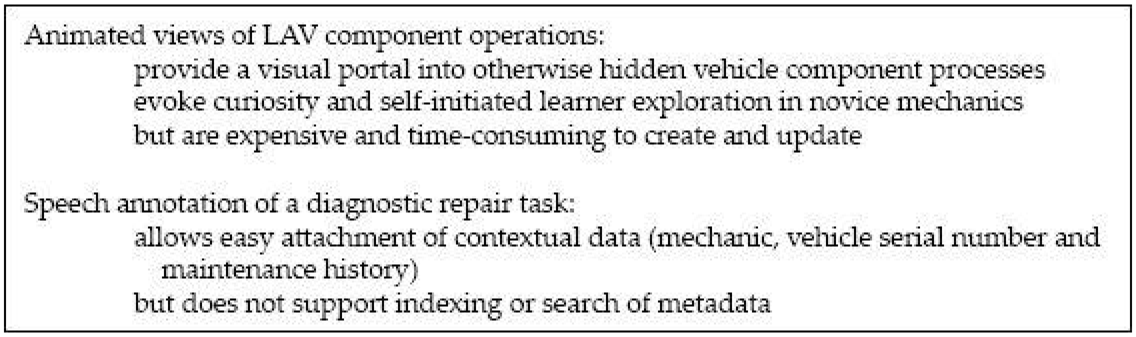
Design Moves have many Effects and Side-Effects: Scenarios have many Views

Scenarios are multifarious design objects; they describe designs at multiple levels of detail and with respect to multiple perspectives. A scenario briefly sketches tasks without committing to details of precisely how the tasks will be carried out or how the system will enable the functionality for those tasks. The portable maintenance device scenario in Figure 7.1 is at an intermediate level, with some detail regarding task flow; it could be elaborated with respect to Sgt. Garcia's moment-to-moment thoughts and experiences in order to provide a more elaborated cognitive view, or with respect to individual interactions to provide a more detailed functional view. Alternatively, the scenario could be presented from the point of view of a novice LAV mechanic watching Sgt. Garcia, perhaps in order to learn how to operate the PMD. It could be elaborated in terms of hardware and software components that could implement the envisioned functionality in order to provide a system view. Each of these variations in resolution and perspective is a permutation of a single underlying use scenario.

Scenarios can leave implicit the underlying causal relationships among the entities in a situation of use. For example, in Figure 7.1 the envisioned speech annotation capability allows adding personal, context-sensitive comments on the diagnostic task without the overheads of opening an editor and typing text. However, the annotation is non-coded, and thus cannot be edited symbolically. These relationships are important to the scenario, but often it is enough to imply them. This is an aspect of the property of being concrete but rough that we discussed above.

Sometimes it is useful to make these relationships explicit. For example, in another scenario Sgt. Garcia may wish to revisit his verbal comments on an earlier repair, one that was particularly complex and difficult. Unfortunately, his non-coded voice annotations cannot be searched by string. Thus, this new scenario would end in failure. To understand and address the variety of desirable and undesirable consequences of the original annotation design move, the designer might want to make explicit the relevant causal relationships in these two scenarios. Doing so provides yet another view of the envisioned situations, as shown in Figure 7.3.

Scenarios help designers manage trade-offs. For example, the data structures for the PMD's mechanic voice annotation functionality might differentiate between annotated and non-annotated items, allowing annotated items to be retrieved and browsed as a subset. This would not allow Sgt. Garcia to directly retrieve the set of items with a particular annotation, but it would still simplify the search. Alternatively, the search problem might be addressed directly by speech recognition or audio matching, or by including the option of text annotation. Each of these alternatives would entrain different elaborations for both the annotation scenario in Figure 7.1 and the search scenario discussed immediately above. These elaborations could then be explored for further consequences and interdependencies.


Animated views of LAV component operations:
provide a visual portal into otherwise hidden vehicle component processes
evoke curiosity and self-initiated learner exploration in novice mechanics
but are expensive and time-consuming to create and update

Speech annotation of a diagnostic repair task:
allows easy attachment of contextual data (mechanic, vehicle serial number and maintenance history)
but does not support indexing or search of metadata

Figure 7.3 Consequences associated with the LAV component animation and the speech annotation capability in the diagnosis and repair task

Scientific Knowledge Lags Design: Scenarios can be Abstracted and Categorized

Scenarios exemplify particular themes and concerns in work and activity situations. Earlier we discussed two scenarios for Marine Corps personal maintenance device. In one (Figure 7.1), a person is at first exploring an information structure, but eventually adopts a particular theory about a diagnosis that guides the rest of his exploration. In the other, a person wishes to search and organize information that has previously been browsed. Described at this level of generality, these are not scenarios unique to personal maintenance devices, or even to computers. They are general patterns for how people work with information. Therefore, it is likely that some of the lessons learned in managing the "guided exploration" pattern or the "searching under a description" pattern in the design of any given situation might be applicable in the subsequent design of other situations. Such a taxonomy of scenarios provides a framework for developing technical design knowledge.

Scenarios can also be classified in terms of the causal relations they comprise. In Figure 7.1, for example, providing speech annotation simplifies the actions needed to personalize and augment a piece of information. In this causal relation, the consequence is the simplification of organizing and categorizing—a general desideratum in designing interactive systems. Generalizing the relation in this way allows the feature associated with the consequence (in this example, speech annotation) to be understood as a potential means for that consequence, and employed to that end in other design contexts. There is of course no guarantee that the generalization is correct, that can only be settled by trying to use it and succeeding or failing. The point is that such candidate generalizations can be developed from scenario descriptions.

The generalization of the causal relations comprising scenarios can also be carried out across features: speech annotation of data helps people create a personalized view of their information, and allows capturing the knowledge gained for undertaking specific tasks (such as an LAV repair) in context. But this relation holds independent of whether the data is annotated by speech, by text or by handwriting. Understanding the relation more generally allows designers to consider any medium for annotation as a potential means of facilitating a personalized data view.

Scenarios can also be taken as exemplars of model scenarios; for example, Figure 7.1 illustrates a model of guided exploration. Sgt. Garcia follows the "advice" of the personal maintenance device only because he is reasonably confident of the accuracy and effectiveness of the device. The system was designed to support this style of use; to that extent it embodies a model of guided exploration. Other models are possible of course; many diagnostic systems would require a maintainer to follow a specified diagnostic tree, even if that meant following branches that an experienced mechanic knows to be irrelevant to the current component fault. Seeing the scenario in Figure 7.1 as a guided exploration scenario allows the designer to benefit from prior knowledge pertaining to this model and to contribute further design knowledge of the model based on the current project.

HOW IS SCENARIO-BASED DESIGN USED?

One of the strengths of scenario-based methods is that while they are centered on a single representation, the scenario, they are able to use this representation in a multitude of ways at different points in the systems development lifecycle.

Requirements Engineering

Requirements engineering is the process of identifying and articulating needs for new technology and applications. It involves a wide range of activities including interviews with potential customers and future users of new systems, observations of workplace activity that new systems are intended to support, and participatory workshops involving direct collaboration among various stakeholder constituencies (users, their managers and co-workers, system developers, documentation designers, user interface specialists).

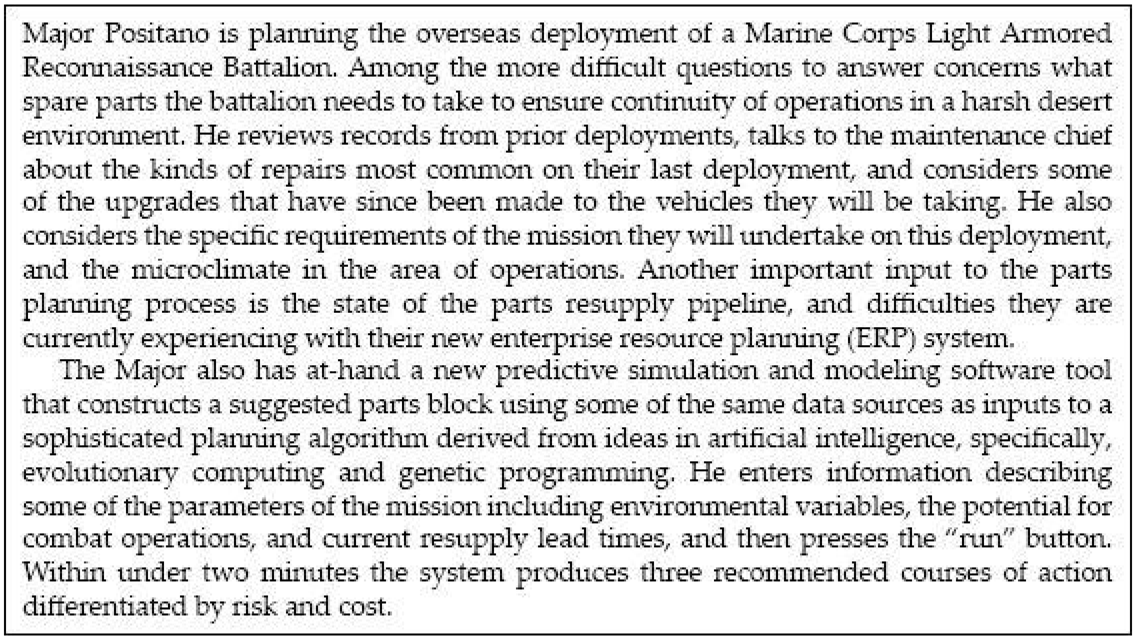
Several kinds of scenarios are used in the requirements process. Routine workflow scenarios describe typical episodes of the human activity that the new system must support. By describing work practices vividly, scenarios help to keep developers focused on the user's needs. Problem scenarios describe episodes of breakdown in workflow, including difficulties in usability and threats to safety. They concretize outcomes that the new system must avoid. Envisionment scenarios describe possible alterations of extant situations. They can be taken as initial design proposals, sometimes called paper prototypes, and at the same time be used instrumentally to evoke further requirements and analysis from users and developers. They encourage "what if?" reasoning.

As an example, a problem scenario might describe obstacles a Marine Corps light armored vehicle commander encounters in determining the parts block (the number and type of spare parts) required for an overseas deployment. A related envisionment scenario might describe how such a task might be carried out using a predictive simulation and modeling tool, as in Figure 7.4.

Scenarios help to integrate the requirements engineering process by providing a single type of representation to evoke, codify, and analyze various sources of requirements information. The trade-offs implicit in the scenarios can be articulated as an a priori design rationale, guiding design work rather than merely describing its outcomes. For example, the new predictive simulation and modeling tool may increase the accuracy of parts planning, as well as reduce planning time and cost, but it may result in over-reliance on technology that may not always be available to the Marine commander.

Such an explicit design rationale can help to clarify the root concepts for the system early in the development process, promoting more elegant design solutions. The accessibility of scenarios to non-technologists facilitates greater participation by users. Scenarios can be created by the users themselves, or created by designers and shared with users to identify and prioritize requirements.

The flexibility of scenarios facilitates management of emergent requirements—the new design goals that often become apparent during system development. Requirements scenarios can be modified continuously to provide an overall view of the most current understanding of the goals for the system.



Major Positano is planning the overseas deployment of a Marine Corps Light Armored Reconnaissance Battalion. Among the more difficult questions to answer concerns what spare parts the battalion needs to take to ensure continuity of operations in a harsh desert environment. He reviews records from prior deployments, talks to the maintenance chief about the kinds of repairs most common on their last deployment, and considers some of the upgrades that have since been made to the vehicles they will be taking. He also considers the specific requirements of the mission they will undertake on this deployment, and the microclimate in the area of operations. Another important input to the parts planning process is the state of the parts resupply pipeline, and difficulties they are currently experiencing with their new enterprise resource planning (ERP) system.

The Major also has at-hand a new predictive simulation and modeling software tool that constructs a suggested parts block using some of the same data sources as inputs to a sophisticated planning algorithm derived from ideas in artificial intelligence, specifically, evolutionary computing and genetic programming. He enters information describing some of the parameters of the mission including environmental variables, the potential for combat operations, and current resupply lead times, and then presses the "run" button. Within under two minutes the system produces three recommended courses of action differentiated by risk and cost.

Figure 7.4 Envisionment Scenario, "Planning An LAR deployment parts block"

Human–Computer Interaction

Human–computer interaction involves the design and evaluation of systems as they are experienced and used by people. It includes understanding and modeling the perception, cognition, behavior, and social interaction of work activity, and developing user interface displays and interaction devices, and associated documentation and training to support human characteristics and preferences.

Human–computer interaction forces a strongly task-oriented perspective on defining systems and their use. Human–computer interaction designers use scenario walkthroughs to develop transparent and consistent rubrics for the appearance and behavior of displays and controls, sometimes called user interface metaphors. Scenario walkthroughs are used to ensure that the information displayed by the system in any given state suggests appropriate actions and interpretations to users, and that error recognition, diagnosis, and recovery information is available, accurate, and easy to employ. Training, online help, and other system documentation now typically presents information in a task context, that is, as a scenario description.

In the system development process, human–computer interaction closely couples design and evaluation. Envisionment scenarios can be detailed in terms of specific goals for user performance and experience, for example, how quickly on average users will be able to perform routine tasks, like scheduling a community meeting, with what error rates, and with what levels of satisfaction. Such explicit usability goals can become part of the design scenarios for the system, and subsequently can be assessed with test subjects.

Such evaluation work goes on throughout the development process. Early in development, the performance and experience of subjects in test scenarios can be used as formative evaluation to refine the original goals for the system. Later in the development process, similar scenarios can be used as summative evaluation to verify that development goals were met.

Object-Oriented Software Engineering

Scenario-based approaches to requirements engineering and human–computer interaction have been dramatically integrated in the past decade by scenario-based approaches to software development. The key bridging insight is that the people and things described in requirements scenarios, and the user interface display objects and controls of human–computer interaction scenarios, can correspond to the same software objects. This insight derives from the distinction between software models and views in object-oriented software: software models describe underlying functionality, views describe the presentation of the functionality.

Scenario-based approaches to object-oriented software engineering suggest taking the nouns and verbs of an envisionment scenario as a first approximation object model: the nouns are the objects, the verbs are the methods (capabilities) of the objects. In the parts block planning example, vehicle (LAV), component, plan, repair, and the operating environment would be part of the initial object model. A vehicle can be repaired, a spare part can be consumed, a plan can be computed, and so on.

Scenarios are then "run" as simulations on such object models, successively elaborating the models in terms of what each object is responsible for (what data it holds, what methods it uses) and how objects collaborate (share data to enable one another's methods). For example, running the community meeting scenario against the initial object model suggests that the meeting object creates the form to gather user data, that it works with the room to schedule itself, that it holds its own topic, date/time, and location, and that it creates the notice.

A stronger scenario-based software development method is use cases. A use case is a set of system event traces that accomplish a particular scenario goal within a given system model. A simple example is the "Get Cash" use case in an ATM machine. The set of event traces gives the software designer a precise way of verifying how the use case scenario has been implemented in the software.

Scenario-Based Evaluation

Scenarios are also useful as test cases for prototypes or production systems. Tradeoffs identified in scenario-based design can later be employed as criteria or metrics in evaluation activities. Especially in cases where scenarios are used early in the development lifecycle, they can serve as a sort of contract between stakeholders and the development team. They couch system requirements in a form and vocabulary that both understand, so can help reduce the ambiguities and miscommunication that can result when more abstract and technical representations are used as the basis for the development of system capabilities.

We evaluate systems framed by the tasks those systems were designed to enhance. Scenario-based methods emphasize this relationship, creating and maintaining task narratives throughout the system development process. Scenarios are available and referred to throughout all phases of the development process, enabling early upstream formative evaluation, coherently evolving into focused and efficient summative evaluation. Even guideline-driven analytic evaluation can be carried out more easily and more thoughtfully if framed by realistic domain tasks. Any kind of empirical evaluation must start with consideration of the tasks that participants will be carrying out. Evaluation task scenarios are a natural place for productive contact between designers and users: the users are engaged by the system as framed in the tasks they understand, perform, and will perform with the system. This is also where the users have critical expertise to share.

The trade-offs we illustrated in Figure 7.3 above analyze scenario narratives into constitutive causal relationships. Such trade-offs are useful criteria for understanding the behavior of people carrying out evaluation scenarios: For example, with respect to Figure 7.3, do participants easily perform and appreciate capabilities for gathering contextual data, do they seem frustrated with respect to search?

Scenarios as a Fulcrum in the System Lifecycle

Scenario-based practices currently 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.

Surveys of contemporary development methods indicate that the use of scenarios is pervasive but not systematic. Developers say they would like more comprehensive methods and better tool support. An underlying tension in this is that a key property of scenarios as planning and design tools is that they seem quite concrete to the analyst/designer, while actually leaving most details unfixed. This has the effect of engaging greater imagination and critical reflection in system development work. It will be interesting to see how the tension between the benefits of roughness and the need for explicit engineering methods is resolved during the next decade.

MOVING FORWARD WITH SCENARIO-BASED DESIGN

Scenario-based design provides a framework for managing the flow of design activity and information in the task-artifact cycle. Scenarios evoke task-oriented reflection in design work; they make human activity the starting point and the standard for design work. Scenarios help designers identify and develop correct problem requirements by being at once concrete and flexible. They help designers to see their work as artifacts-in-use, and through this focus to manage external constraints in the design process. Scenarios help designers analyze the varied possibilities afforded by their designs through many alternative views of usage situations. And scenarios help designers cumulate their knowledge and experience in a rubric of task-oriented abstractions. This is depicted in Figure 7.5.

In current practice, scenario-based design techniques are widely used in human–computer interaction. Many new methods for capturing, generating, and classifying scenarios are emerging. For example, scenarios can be directly negotiated with end-users (participatory design), generated by systematically instantiating the signature phenomena of theories of human activity (theory-based design), or by various heuristic brainstorming techniques, like question generation and analogy. Scenarios can be applied in a wide variety of ways in system development. Besides their direct roles as descriptions of current or envisioned system use, they can be employed as bounding contexts for developing design rationales, for theories of human–computer interaction, and for software development (for example, as use cases).

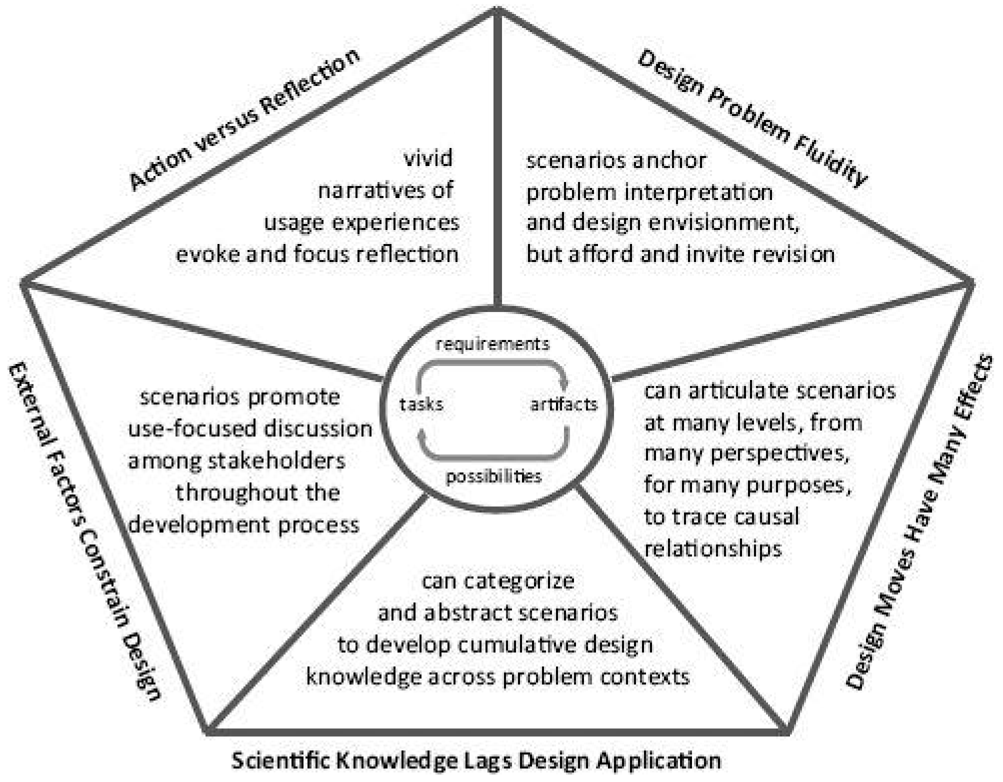

Action versus Reflection
Design Problem Fluidity
vivid narratives of usage experiences evoke and focus reflection
scenarios anchor problem interpretation and design envisionment, but afford and invite revision
requirements
tasks
artifacts
possibilities
scenarios promote use-focused discussion among stakeholders throughout the development process
can articulate scenarios at many levels, from many perspectives, for many purposes, to trace causal relationships
can categorize and abstract scenarios to develop cumulative design knowledge across problem contexts
External Factors Constrain Design
Design Moves Have Many Effects
Scientific Knowledge Lags Design Application

Figure 7.5 Challenges and approaches in scenario-based design

A key challenge is to more systematically leverage this wide variety of scenario-based activity within the development process. The most promising recent developments involve closer integration of practice in human–computer interaction with the development of standard schemata, and support for libraries and reuse frameworks developed in requirements engineering.

REFERENCES

Carroll, J.M. (Ed.) 1995. *Scenario-based design: Envisioning work and technology in system development*. New York: John Wiley and Sons.
Carroll, J.M. 2000. *Making use: Scenario-based design of human–computer interactions*. Cambridge, MA: MIT Press.
Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. 1992. *Object-oriented software engineering: A use case driven approach*. Reading, MA: Addison-Wesley.
Jarke, M., Bui, X.T., and Carroll, J.M. 1998. Scenario management: An interdisciplinary approach. *Requirements Engineering*, 3(3–4), 155–173.

8

Socio-Cognitive Issues in Human-Centered Design for the Real World

Saadi Lahlou

In order to avoid resistance and hidden costs in the deployment and maintenance phase of complex socio-technical systems, we developed a participative technique which addresses the deployment and maintenance issues early in the design process: “experimental reality.” It enables realistic stakeholders’ involvement at an early stage of design, by developing the new system in actual context of use. This approach implies a different way of managing development projects, and in contrast highlights some shortcomings of current practice.

This chapter provides; (1) a framework, installation theory, to sort out and address the problems encountered by design projects for complex socio-technical systems, (2) a quick presentation of activity theories, and (3) an illustration of our design technique in the domain of information technology systems supporting collaborative work in a large industrial organization.

1. INTRODUCTION: DESIGNING FOR REAL-WORLD SYSTEMS

Although we design for the future, this future is supposed to start at the end of the design project. This is usually pretty close, and by the time the new system starts it will still have to be compatible with a lot of the current world-as-is-now. This chapter addresses this issue.

The world in which we design comes with some already installed basis: physical (equipment, devices), social (laws, customs, norms), and cognitive (habits, education). Even the users are “second-hand”: they are not new to the system being designed in the sense that they have already been educated and trained within the current system. The designer will have to cope with this pre-existing installation. *Installation theory* (section 1) provides a framework for comprehensive design and a checklist of the three levels of reality which the designer should address.

If we want to design in a user-centric way, we must be on the user’s side. But observing what users do is not enough. Current behavior of the users is a biased indication of what the users want, precisely because current behavior is framed by present installation. User-centric design should focus on what the users actually want: their motives, their goals. We found Russian Activity Theory to be the most efficient for complex system design among the variety of theories we tried. Section 2 presents a remix of this theory, adapted for design purposes with some additions of current distributed cognition and psychology.

In section 3 we address the case of designing Information Technology (IT) systems. These are specific because they are by nature communicating with the rest-of-the-world–hence they must adapt to the fast-changing technological context. The need to support openness and connectivity confronts their design with the Sisyphean “never-endingness” of continuous upgrade. We present a design technique, “experimental reality,” tailored for this problem. This is illustrated by the example of conference rooms.

2. DESIGN AND CHANGE MANAGEMENT: MAKING NEW IN THE OLD CONTEXT

Designers of a single object or service may (sometimes) find it possible to draw the limit of the problem-space they should address and work with a clear set of specifications to redesign on a blank sheet. But designers of large socio-technical systems are often cornered into “upgrading” a previous system and keeping some continuity with the current state of things. This is because of the complexity (relations between parts of the system) and the impossibility of the redesign of some of the parts. For example, a new IT system, a new transportation system, a new plant and even a new building must take for granted a series of limiting socio-technical specifications. They must fit into the existing frameworks of the environment which surrounds them (the organization, the transportation network, the trade, the city, and so on). Designers must then cope with existing users, installed basis, and existing rules.

Subsection 2.1 introduces installation theory taking inspiration on how the global society deals with these issues to generate some design guidelines. Subsection 2.2 provides some indications to involve users in a realistic design process.

2.1 Installation Theory

This section introduces “installation theory,” a general framework describing the evolution of socio-technical systems. Although installation theory is general in scope, it is useful for practical applications as it clearly delineates three levels where action should by taken to ensure acceptability of change, smooth operation, and future evolution. These levels are:

- the physical level of artifacts where we design affordances;
- the psychological level of representations and practices where users are trained to acquire adapted competence;