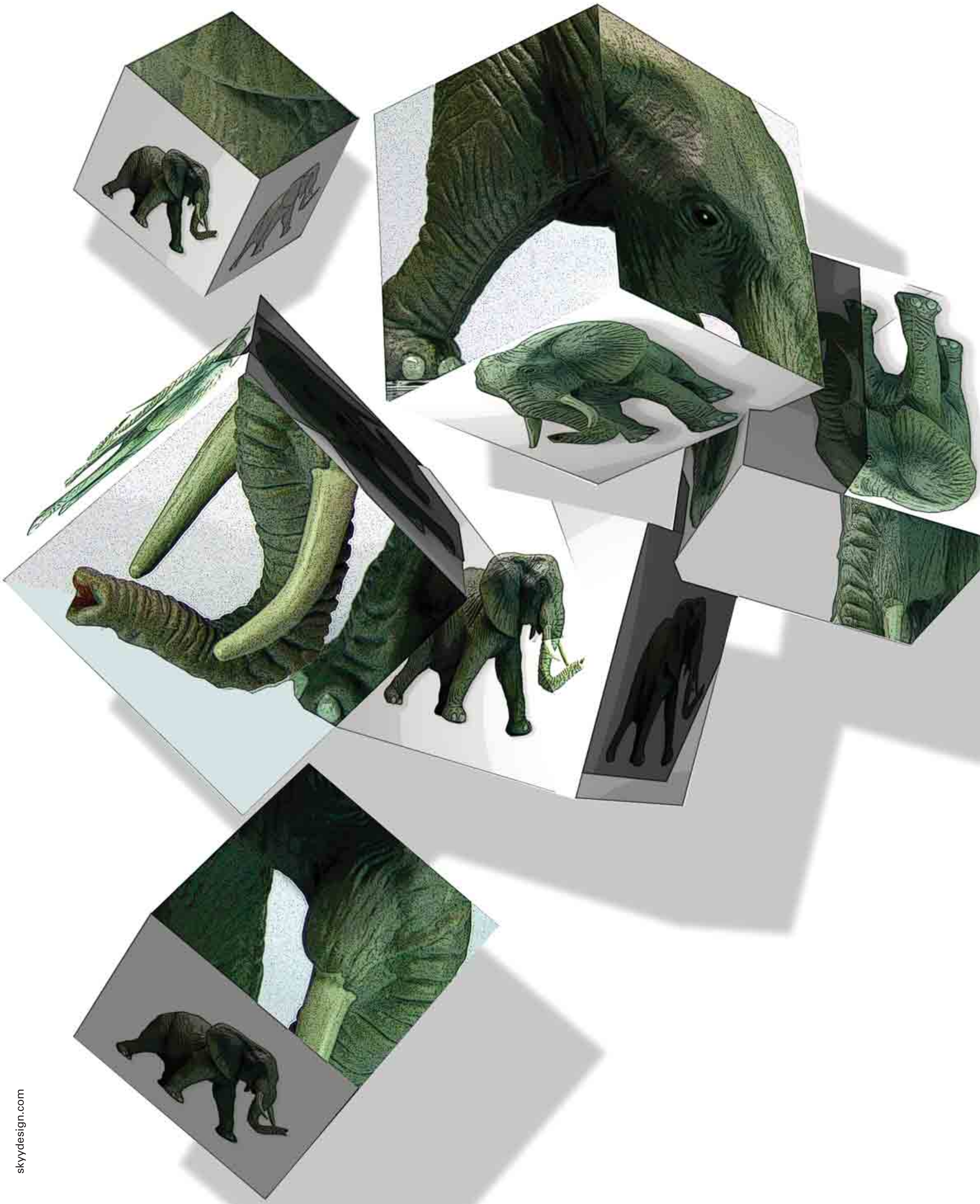

skydesign.com

The Blind Men and the Elephant: Views of Scenario-Based System Design

By Kentaro Go
Center for Integrated Information Processing
University of Yamanashi
Kofu 400-8511 Japan
go@yamanashi.ac.jp

By John M. Carroll
School of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802 USA
jmcarroll@psu.edu

Six blind men encounter an elephant. Each of them touches a different part of the elephant and expresses what the elephant is. Although they are touching the same elephant, each man's description is completely different from that of the others. We have been using this story as a metaphor for understanding different views of scenario-based system design.

Scenario-based system design is an approach that employs scenarios as a central representation throughout the entire system lifecycle. The approach encourages user involvement in system design, provides shared vocabulary among the people participating in the system development project, envisions the uncertain future tasks of the system users, and enhances ease of developing instructional materials. It provides a good brainstorming tool for planning and allows the stakeholders to consider alternative choices in decision-making. It addresses dynamic, multiple, parallel, and/or distributed factors in a manageable manner. This rich variety of roles is selectively used from different viewpoints in diverse communities including human-computer interaction, strategic planning, requirements engineering, and object-oriented analysis/design. It is not easy to see what *the elephant* actually is.

The purpose of this paper is to help build a common language in software design for stakeholders from a wide range of disciplinary backgrounds providing an overview of scenario-based system design. By surveying the history and typical scenario usage in different fields, we demonstrate the importance of scenario-based approaches in system development. Human-computer interaction can be more effective if it can cooperate with and leverage efforts in other fields. Likewise, human-computer interaction can more effectively contribute to the evolution of the other fields. Perhaps scenarios offer an opportunity for this mutual contribution.

In addition, the history of scenario use provides a significant message. The tendency in the past 30 years is to look at finer grains of description (year-in-the-life scenario in strategic planning to day-in-the-life scenario in requirements engineering, to moment-to-moment scenario in human-computer interaction and object-oriented analysis/design). We suggest that now is the time to consider reintegrating the field by backing up the tree—that is, taking a moment-to-moment scenario and integrating it with a day-in-the-life scenario, and so forth. Indeed, current trends in application development that employ use scenarios with strategic planning are the beginning of this reintegration.

Marissa was not satisfied with her class today on gravitation and planetary motion. She is not certain whether smaller planets always move faster, or how a larger or denser sun would alter the possibilities for solar systems.

She stays after class to speak with Ms. Gould, but she isn't able to pose these questions clearly, so Ms. Gould suggests that she re-read the text and promises more discussion tomorrow.

Figure 1. An example narrative scenario from Rosson and Carroll [19]

Scenarios in System Design

A scenario is a description that contains (1) actors, (2) background information on the actors and assumptions about their environment, (3) actors' goals or objectives, and (4) sequences of actions and events. Some applications may omit one of the elements or they may simply or implicitly express it. Although, in general, the elements of scenarios are the same in any field, the use of scenarios is quite different.

Scenarios are expressed in various media and forms. For example, scenarios can be textual narratives, storyboards, video mockups, or scripted prototypes. In addition, they may be in formal, semi-formal, or informal notation. A typical example of an informal scenario is a story (Figure 1), a kind of scenario frequently used for envisioning user tasks in human-computer interaction.

We will examine four communities that actively use scenario-based approaches: strategic planning, human-computer interaction, requirements engineering, and object-oriented analysis/design. Strategic planning forecasts the future environment of an organization and helps stakeholders plan actions. Human-computer interaction aims to design usable computer systems that support their users' tasks in a safe and fluent manner. Requirements engineering is about eliciting users' needs regarding computer systems, and about producing specifications; the specifications must be unambiguous, consistent, and complete. Object-oriented analysis/design is a methodology for constructing a world model; the model is based on the idea of objects associating with data structure, class hierarchy, and object behavior. The scenario-based approaches of the four communities are summarized in Table 1, Table 2, and Table 3, and a history of scenario-related fields is summarized in Figure 2.

Strategic Planning

The discussion of scenario usage in computer system design is relatively new, but scenario-based approaches in planning and management have quite a long history. In his book *Thinking About the Unthinkable*, published in 1962, Herman Kahn considers scenarios as an aid to thought in uncertainty [14]. Kahn uses scenarios (1) as an analysis tool, (2) in narrative form, and (3) with psychological, social, and political assessments. He encourages combining role-playing exercises with scenario development. He says, "A scenario results from an attempt to describe in more or less detail some hypothetical sequence of events. Scenarios can emphasize different aspects of 'future history.'" He goes on to say, "The scenario is particularly suited to dealing with several aspects of a problem more or less simultaneously. By the use of a relatively extensive scenario, the analyst may be able to get a feel for events and the branching points dependent upon critical choices. These branches can then be explored more or less systematically." He then continues, "The scenario is an aid to the imagination."

He lists five advantages of the scenario as an aid to thinking:

1. Scenarios make the analyst salient to important events that should be taken into account in the uncertain future.
2. Scenarios require the treatment of details and dynamics.
3. Scenarios provide assistance in articulating the interaction of psychological, social, political, and military aspects, including the individual influence.

4. Scenarios can illustrate certain principles or questions.
5. Scenarios can be used to reason about alternative possibilities for past or present crises.

From a historical viewpoint, one of the milestone papers in the field is Pierre Wack's, which describes how in the late 1960s and early 1970s, the Royal Dutch/Shell Group constructed scenario-planning techniques to foresee and prepare for the 1973 oil crisis [20]. He distinguishes two ways to employ scenarios. On the one hand, scenarios are used for direct *forecasting*, such as the case with Royal Dutch/Shell. On the other hand, scenarios can be used for *planning*, since current scenarios can lend to brainstorming about the future. Wack discusses scenario planning as an approach that helps stakeholders think about new possibilities. He emphasizes the iterative construction of scenarios, in which new scenarios are derived through the analysis of the old ones.

After Wack's paper, scenario use in strategic planning focused on the second use, that is, as an analysis tool. In 1992, *Planning Review* had special issues on scenarios in strategic management.

In summary, the dominant idea in the

COMMUNITY	ACTOR	ENVIRONMENT
Strategic Planning	Specific organization CEO as in organizational role Planning organization may not be main actor	Other political, economical entities
Human-Computer Interaction	Specific user Has internal states	System Workplace contexts
Requirements Engineering	General user System Describes actor's behavior	System
Object-Oriented Analysis/ Design	General user System Describes actor's behavior	System

Table 1. Major characterizing elements of the scenario-based design approaches

COMMUNITY	SCENARIO USAGE
Strategic Planning	Envisioning uncertain future environment Providing communication tool Organizational learning Sharing a mental model among stakeholders
Human-Computer Interaction	Analyzing user tasks Envisioning future work Mock up and prototyping Evaluating the constructed system Deriving learning materials Developing design rationale
Requirements Engineering	Eliciting user requirements Deriving specifications Analyzing the current system usage Describing the current system usage Constructing test cases
Object-Oriented Analysis/ Design	Modeling objects, data structures and class hierarchy Analyzing problem domain Providing a model of real-world objects

Table 2. Typical scenario usage in design

COMMUNITY	UNCERTAIN FACTOR	GOAL OF SCENARIO-BASED APPROACH	DEVELOPMENT PROCESS	VIEW POINT	BACKGROUND GOAL OF COMMUNITY
Strategic Planning	Environment	List "what-if" questions and their answers	Iterative	Organization Technological changes Economics Social, political regulations Consumer attitudes	Plan a course of actions
Human-Computer Interaction	Use of system	Envision user requirements of (future) system use	Iterative Prototyping	Human Usability Cognition Emotion	Describe use of (future) systems Design usable computer system
Requirements Engineering	System requirements Functionality	Acquire user requirements and specify them	Waterfall Spiral	System architecture Development process	Specify systems Provide a good transition to the next development phase
Object-Oriented Analysis/ Design	Objects Data structures Class hierarchy	Identify objects, data structures and model class hierarchy	Iterative Incremental	System Object	Design a model of world

Table 3. Some factors that can be used to categorize scenario usage

COMMUNITY	1960s	1970s	1980s	1990s
Strategic Planning	Kahn's book, 1962 Royal Dutch/Shell scenario planning, c. 1965 [20]		Wack's paper, 1985 [20]	Planning Review's special issues, 1992
Human-Computer Interaction			diSessa, 1985 [7] Olympic Message System, 1987 [8]	SIGCHI bulletin's discussion, 1992 Carroll's book, 1995 [2]
Requirements Engineering			Hooper and Hsia, 1982 [9]	Potts et al., 1994 [17] Hsia et al., 1994 [10] CREWS' classification, 1996 [18]
Object-Oriented Analysis/Design			Use-case approach, 1987 [13]	Responsibility-driven approach, 1990 [22] Koskimies et al., 1997 [15]

Figure 2. A historical contrast of scenario usage and discussion in strategic planning, human-computer interaction, requirements engineering, and object-oriented analysis/design

strategic planning community is that scenario planning is a process of envisioning and critiquing multiple possible futures.

Human-Computer Interaction

Human-computer interaction is another field that actively discusses what scenarios are and how to use them in system design [1], [2], [5], [23]. Human-computer interaction uses scenarios to describe the use of systems and to envision more usable computer systems. To observe and then analyze the current usage of a system, it is necessary to involve authentic users. In this approach, actors in a scenario are specific people who carry out real or realistic tasks. To envision the use of a system that has not yet been constructed, the scenario writers have to describe potential users and what they may do with the system in extensive detail, including, for example, a description of workplace contexts.

Day-in-the-life scenarios are one of the most powerful methods for envisioning authentic computer use. They illustrate users' daily activity with computers over time. For example, a scenario might describe how a musician uses music software through various input devices and gets frustrated [16]. Day-in-the-life scenarios can be used to envision the future use of computers. One famous example is Apple Computer's "Knowledge Navigator" video [6]; it shows how a person interacts with computer capabilities that have not yet been developed. The scenario gives a vivid view of how people could use computers as an intelligent assistant in daily life.

Envisioned scenarios can be analyzed to create explicit rationale for future designs [4]. Carroll and Rosson proposed an iterative process for writing scenarios and analyzing their psychological claims. They produced textural narrative scenarios for envisioning future use of a system; then, they conducted claims analysis by listing the positive and negative consequences of features of tools and artifacts in the scenarios. After examining the claims, they derived new scenarios. This iterative process makes the design of the system more precise at every stage.

Evaluation of systems is another way scenarios are used in human-computer interaction. During the evaluation process, careful observations of a real work setting are necessary. The technique is frequently used in human-computer interaction to analyze social aspects of user tasks. Scenarios are used in ethnographic field study as a device for describing the context of work. Scenarios are then analyzed to reveal how the work is socially organized [11].

One of the significant views derived from the study of human-computer interaction is that scenarios are not specifications. Carroll contrasts two complementary perspectives: The perspectives clearly separate scenarios from specifications [2]. Scenarios are (1) concrete descriptions, (2) focus on particular instances, (3) work driven, (4) open-ended, fragmentary, (5) informal, rough, colloquial, and (6) envisioned outcomes. In contrast, specifications are (1) abstract descriptions, (2) focus on generic types, (3) technology driven, (4) complete, exhaustive, (5) formal, rigorous, and (6) specified outcomes.

The earliest scenario usage in human-computer interaction originates in the mid-1980s, and is cited by diSessa, and Gould, Boies, Levy, Richards, and Schoonard [7], [8]. diSessa explains the Boxer programming environment by using scenarios. He did not directly employ scenarios in the programming environment design but tried instead to explain the design by using scenarios. Gould and his colleagues' work did make the direct use of scenarios in system design; they used the scenario dialogues to design the 1984 Olympic Message System. There were discussions of scenarios in *SIGCHI Bulletin* in 1992, and an edited volume of scenario-based design was published in 1995 [2].

Requirements Engineering

Because the goal of requirements engineering is to elicit and specify users' requirements, its scenario usage focuses on analysis. In particular, it gives weight to how to specify the requirements and provide a smooth transition to the next development phase. Scenarios for this purpose, therefore, must be written from the system's viewpoint. This makes scenario-based requirements engineering relatively concrete and process-oriented as a methodology.

In requirements engineering, one of the earliest works is Hooper and Hsia [9]; they proposed the idea of scenarios as prototypes to identify user requirements. "Prototyping is a 'quick and dirty' construction of a system (or part of a system). In prototyping by use of scenarios, one does not necessarily model the system or any component thereof directly, but rather represents the performance of the system for selected sequences of events." This approach can be simpler than the whole system modeling. By using scenarios, the users can simulate the real operation of a system and know their actual needs.

Typical scenario-based approaches in requirements engineering include the formal scenario analysis by Hsia, Samuel, Gao, Kung, Toyoshima, and Chen [10], the inquiry-based requirement analysis by Potts, Takahashi, and Anton [17], and the CREWS project [18].

Hsia and associates [10] proposed a formal approach to scenario analysis, a requirement analysis model in the early phases in software development. Their approach defines systematic development stages from the initial semi-formal scenarios to the final formal scenarios. Each of the stages except the first use scenarios in formal notation, which enable the system analysts to derive part of the system-requirement specification.

Potts and colleagues [17] developed the Inquiry Cycle Model of requirement analysis, which is "a structure for describing and supporting discussions about system requirement." The model consists of three phases of requirements: documentation, discussion, and evolution. These three phases make a cycle for acquiring and modeling the knowledge of problem domain.

Their scenarios are part of the requirements documentation, which are in hypertext form. The scenarios are intended to be semi-formal; in fact, they are expressed in tabular notation. "In the broad sense, a scenario is simply a proposed specific use of the system. More specifically, a scenario is a description of one or more end-to-end transactions involving the required system and its environment."

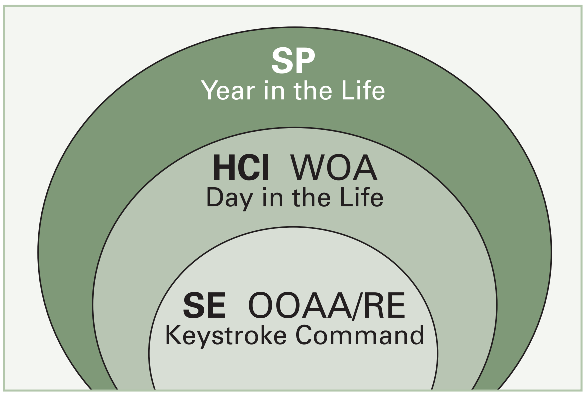

Figure 3. The nested structure of Strategic Planning (SP), Human-Computer Interaction (HCI), and Software Engineering (SE). HCI includes Work-Oriented Approach (WOA), and SE consists of Object-Oriented Analysis/Design (OOAD) and Requirements Engineering (RE).

The CREWS project (Cooperative Requirements Engineering with Scenarios) is a large European project on scenario use in requirements engineering [18]. Based on the existing techniques and tools, the project is trying to make scenario usage a systematic engineering discipline; hence, its approach to scenario usage is tool- and/or method-driven. The project has specific goals. With the multimedia-recorded system use, it provides a common medium for discussing system requirements among multiple stakeholders. With natural language understanding, it provides a device to elicit requirements in a graphical representation. By cooperative requirement animation and by comparison between a specification and its test scenarios, it expedites the validation process of a system.

In the CREWS project, Rolland, Achour, Cauvet, Ralyte, Sutcliffe, Maiden, Jarke, Haumer, Pohl, Dubois, and Heymans [18] have developed a framework to classify scenarios in scenario-based approaches. They give four dimensions: the form view, the contents view, the purpose view, and the lifecycle view. The form view represents the format of scenarios; for example, narrative texts, graphics, images, videos, and prototypes are in the form view. In addition, formality (e.g., formal, semi-formal, and informal) is discussed from this viewpoint. The contents view expresses what knowledge scenarios express. It consists of four elements: abstraction, context, argumentation, and coverage. In the purpose view, scenarios are categorized from the reasons of usage. The lifecycle view deals with how scenarios are handled (e.g., creation, refinement, and deletion). Applying the framework to eleven scenario-based approaches, they found out that scenarios are used to describe system behaviors interacting with its environment and most of them are in textual representation. They pointed out the importance of the formalization of scenario-based approaches, the variety of the application fields, and the need for practical scenario evaluation.

Object-Oriented Analysis/Design

Object-oriented analysis/design models an application domain. It identifies objects, data structures, and class hierarchies. Its viewpoint is that of a system model.

There are three typical scenario-based approaches: Jacobson's use-case approach [13], Wirfs-Brock's responsibility-driven approach [22], and Koskimies, Systa, Tuomi, and Mannisto's automated modeling support approach [17].

According to Jacobson, a use-case depicts how a user or another system uses a system. A basic concept behind the use-case approach is that a system is well described from a black-box view. A use-case model consists of two elements: actors and use-cases. In essence, "The actors represent what interacts with the system. They represent everything that needs to exchange information with the system. Since the actors represent what is outside the system, we do not describe them in detail [12].”

Jacobson also claims that a use-case is fundamentally different from a scenario: Scenarios correspond to use-case instances. There is no correspondence to use-case classes. A use-case expresses all the possible paths of events, but a scenario describes part of the possible paths. In addition, a use-case seeks a formal treatment defining a model while a scenario seeks an informal treatment [12].

Another object-oriented approach relating to scenarios is the responsibility-driven approach by Wirfs-Brock [22]. She states that the approach emphasizes informal methods for characterizing objects, their roles, responsibilities, and interactions. By using the informal methods, the designer is to determine the software system, stereotype the actors, determine system use-cases, construct conversations, identify candidate objects, identify responsibilities of candidate objects, design collaborations, design class hierarchies, fully specify classes, and design subsystems.

The approach uses cards as tools; they are called “CRC cards” (Class-Responsibility-Collaboration cards). Each of the cards lists responsibilities and collaborations with other objects on the front side; the back side has initial notions in the roles of that object and its stereotypes.

Koskimies and his colleagues proposed automated support for modeling object-oriented software [15]. They follow the definition of scenarios in the Object Modeling Technique (OMT), a commonly used object-oriented analysis/design method: “In OMT, scenarios are informal descriptions of sequences of events occurring during a particular execution of a system.” In fact, Koskimies and his colleagues intended to formalize scenarios in order to provide an automated support of scenario-based system development; therefore, they formalized them as event-trace diagrams.

In summary, each of the four communities has its own use of scenarios. Strategic planning involves lists of “what-if” questions followed by answers that comprise a scenario for action. Human-computer interaction uses scenarios to analyze a system and to envision a more usable system. Requirements engineering determines user and system needs and produces specifications *vis-a-vis* use scenarios. Object-oriented analysis/design scenario usage involves identifying objects and data structures and modeling a class hierarchy. There are clear similarities. Strategic planning and human-computer interaction exploit scenarios to envision future use, actions, and events. The analysis use of scenarios is common to human-computer interaction, requirements engineering, and object-oriented analysis/design. However, instead of seeing local relationships, we will discuss global relationships highlighting the insight of each community.

Relationship of the Four Communities

The relationship among the four communities can be discussed from their structure and lifecycle.

Structure of the Four Communities

The four communities-human-computer interaction, strategic planning, requirements engineering, and object-oriented analysis/design-have a nested relation, in which the latter two communities can be categorized into software engineering (see Figure 3).

From inside to outside, there are three layers: software engineering, human-computer interaction, and strategic planning. Human-computer interaction incorporates work-oriented approaches focusing on users’ tasks and user participation.

The nested structure describes the focus of each field based on the degree of tangibility of the target content of scenarios. Software engineering scenarios focus on real world objects or physical artifacts; furthermore, they include scenarios of the existing system use. Therefore in the field, the materiality of a system attracts great attention for designers and analysts. In addition to tangible artifacts, human-computer interaction scenarios treat user tasks. User tasks are not easily touched. That is, their degree of tangibility is much lower than software engineering. Strategic planning deals with much more abstracted artifacts, such as future plans of an organization. It includes, in some degree, current technology as a basic assumption for planning.

Software engineering scenarios are relatively small in scope. They typically include keystroke- and command-level scenarios. Human-computer interaction scenarios are larger in scope; they deal with day-in-the-life scenarios. Strategic planning scenarios have the widest scope of the three groups; they treat events and issues of the grain of *year-in-the-life*.

The nested structure in Figure 3 also summarizes the history of human-computer interaction. Human-computer interaction—as it originated in software psychology—studied human aspects in traditional system design approach and the use of systems constructed from that approach [3]. It grew out of software engineering research and has focused on practice. More recently, its attention has shifted to work-oriented approaches with researchers recognizing that real work situations are social in nature, and that previous analysis focused only on a single user and single computer interaction. This shift of concept produced a new field: Computer-Supported Cooperative Work (CSCW). CSCW is about work-oriented system design, especially weighted on social and organizational aspects, an area that intersects with strategic planning.

The transition of research interests of design from concrete, physical artifacts to abstract, indefinite organizations of people is clear from a history of these fields relating to scenario-based design.

Lifecycle of Scenarios

The four communities make different use of scenarios in a system lifecycle; indeed, they assume a different lifecycle generally. Strategic planning and human-computer interaction presume development processes as complex, dynamic processes; therefore, they prefer to employ iterative design of scenarios. In strategic planning, Wack described a cyclic process of scenario development [20]. Similarly, in

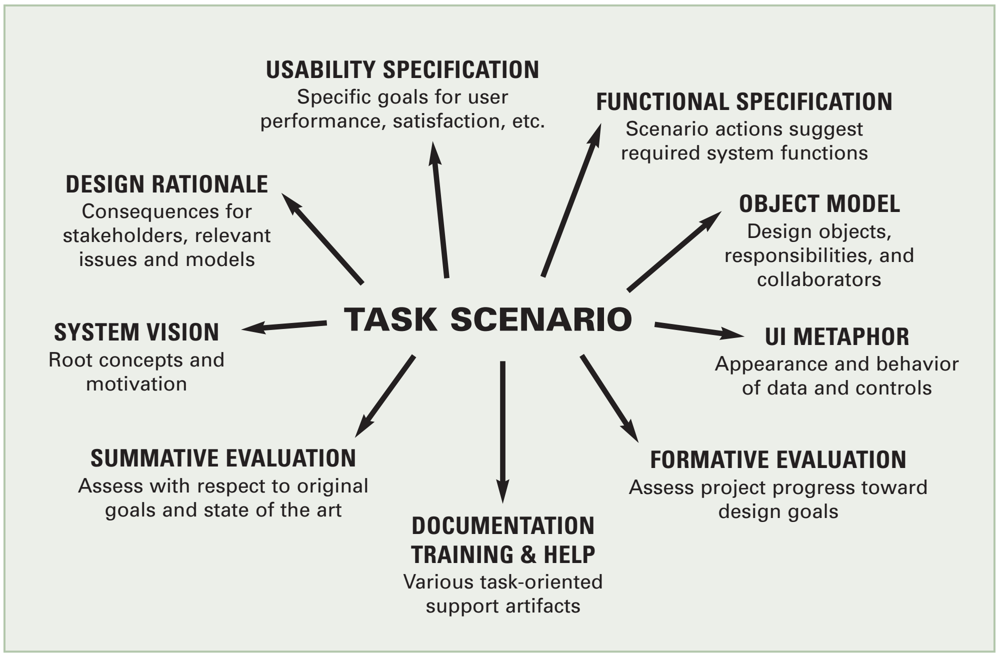
USABILITY SPECIFICATION
Specific goals for user performance, satisfaction, etc.
FUNCTIONAL SPECIFICATION
Scenario actions suggest required system functions
DESIGN RATIONALE
Consequences for stakeholders, relevant issues and models
OBJECT MODEL
Design objects, responsibilities, and collaborators
SYSTEM VISION
Root concepts and motivation
TASK SCENARIO
UI METAPHOR
Appearance and behavior of data and controls
SUMMATIVE EVALUATION
Assess with respect to original goals and state of the art
FORMATIVE EVALUATION
Assess project progress toward design goals
DOCUMENTATION TRAINING & HELP
Various task-oriented support artifacts

Figure 4. Scenarios provide a common language for design.

human-computer interaction, the dominant view is that of an iterative process of design. The requirements engineering community generally assumes that development processes are decomposable; thus, techniques for deriving specifications from scenarios at the end of the requirement acquisition phase become a fundamental issue. The object-oriented analysis/design community tends to apply the object-oriented approach to any phase in the development process; it emphasizes the seamlessness between analysis and design in development using incremental and iterative processes. This discipline can be applied to human-computer interaction and requirements engineering.

Scenarios as Common Language

Scenario-based techniques contextualize and concretize design thinking about people and technologies. Strategic planning scenarios capture organizational context, surprise-free continuations, and the status quo, as well as alternative, "what-if" scenarios. Requirements engineering scenarios help elicit and refine functional descriptions of future systems. Human-computer interaction scenarios help users and developers describe and evaluate technology currently in use and envision activities that new technology could enable. Object-oriented analysis/design use-cases of the system identify the possible event sequences of the system to accurately model the domain objects, data structures, and behaviors. These different applications of scenarios emphasize different viewpoints and different resolutions of detail, and they address different purposes. But the vocabulary of concrete narratives is accessible to and sharable by diverse stakeholders in a design project: planners and managers, requirements engineers, software developers, customer representatives, human-computer interaction designers, and the users themselves. In this sense, scenarios provide a common language for design.

ACKNOWLEDGMENTS

We would like to acknowledge the Dagstuhl Symposium on Scenario Management, Schloss Dagstuhl Internationale Begenungs und Forschungszentrum fur Informatik, February 9-13, 1999, Saarbrucken, Germany, and its principal organizer, Matthias Jarke, for suggesting the direction of this paper. We thank Alistair Sutcliffe, Colette Rolland, Pei Hsia, Annie I. Anton, Lewis Erskine, and Craig Ganoe for the comments on the paper.

REFERENCES

1. Carey, T., McKerlie, D., Bubie, W., & Wilson, J. (1991). Communicating human factors expertise through design rationales and scenarios. In D. Diaper & N. Hammond (Eds.), *People and Computers VI: Proceedings of the HCI '91 Conference*, August 20-23, Edinburgh, UK (pp. 117-130). Cambridge, UK: Cambridge University Press.
2. Carroll, J. M. (Ed.). (1995). *Scenario-based design: Envisioning work and technology in system development*. New York: John Wiley & Sons.
3. Carroll, J. M. (1997). Human-computer interaction: Psychology as a science of design. *International Journal of Human-Computer Studies*, 46, 501-522.
4. Carroll, J. M. (2000). *Making use: Scenario-based design of human-computer interactions*. Cambridge, MA: MIT Press.
5. Clarke, L. (1991). The use of scenarios by user interface designers. In D. Diaper and N. Hammond (Eds.), *People and Computers VI: Proceedings of the HCI '91 Conference*, August 20-23, Edinburgh, UK (pp. 103-115). Cambridge, UK: Cambridge University Press.
6. Dubberly, H. & Mitch, D. (1987). *The Knowledge Navigator*. Apple Computer, videotape. Appears in B. A. Myers ed. HCI'92 Special Video Program.
7. diSessa, A. (1985). A principled design for an integrated computational environment. *Human-Computer Interaction 1*(1), 1-47.
8. Gould J. D., Boies, S. J., Levy, S., Richards, J. T., & Schoonard, J. (1997). The 1984 Olympic message system: A test of behavioral principle of system design. *Communications of the ACM 30*(9), 758-769.
9. Hooper, J. W. & Hsia, P. (1982). Scenario-based prototyping for requirements identification. *Software Engineering Notes 7*(5), 88-93.
10. Hsia, P., Samuel, J., Gao, J., Kung, D., Toyoshima, Y., & Chen, C. (1994). Formal approach to scenario analysis. *IEEE Software 11*(3), 33-41.
11. Hughes, J. A., Randall, D., & Shapiro, D. (1992). Faltering from ethnography to design. In *Proceedings of CSCW'92 Conference*, Oct. 31-Nov. 4, Toronto, Canada (pp. 115-122). New York: ACM.
12. Jacobson, I. (1995). The use-case construct in object-oriented software engineering. In J. M. Carroll (Ed.), *Scenario-based design: Envisioning work and technology in system development*, 309-336. New York: John Wiley & Sons.
13. Jacobson, I., Christersson, M., Jonsson, P., & Overgaard, G. (1992). *Object-Oriented Software Engineering: A use-case driven approach*. Reading, MA: Addison-Wesley.
14. Kahn, H. (1962). *Thinking about the unthinkable*. New York: Horizon Press.
15. Koskimies, K. Systa, T., Tuomi, J., & Mannisto, T. (1998). Automated support for modeling OO software. *IEEE Software 15*(1), 87-94.
16. Mountford, S. J. (1991). A day in the life of ... (Panel). In *Proceedings of CHI'91*, April 27-May 2, New Orleans, LA. (pp. 385-388). New York: ACM.
17. Potts, C., Takahashi, K., & Anton, A. I. (1994). Inquiry-based requirements analysis. *IEEE Software 11*(2), 21-32.
18. Rolland, C., Achour, C. B., Cauvet, C., Ralyte, J., Sutcliffe, A., Maiden, N. A. M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., & Heymans, P. (1996). A proposal for a scenario classification framework. *Requirements Engineering 3*(1), 23-47.
19. Rosson, M. B. & Carroll, J. M. (2001). *Usability engineering: scenario-based development of human computer interaction*. Redwood City, CA: Morgan Kaufmann.
20. Wack, P. (1985). Scenarios: Uncharted waters ahead. *Harvard Business Review 63*(5), 72-89.
21. Weidenhaupt, K., Pohl, K., Jarke, M., & Haumer, P. (1998). Scenarios in system development: Current practice. *IEEE Software 15*(2), 34-45.
22. Wirfs-Brock, R., Wilkerson, B., & Wiener, L. (1990). *Designing object-oriented software*. Englewood Cliffs, NJ: Prentice Hall.
23. Young, R. M. & Barnard, P. J. (1991). Signature tasks and paradigm tasks: new wrinkles on the scenarios methodology. In D. Diaper and N. Hammond (Eds.), *People and Computers VI: Proceedings of the HCI '91 Conference*, August 20-23, Edinburgh, UK (pp. 91-101). Cambridge, UK: Cambridge University Press.