ヒューマンインタフェース 87－8
（2000．1．28）

シナリオにもとづく設計法の現状分析

郷 健太郎* John M. Carroll† 今宮淳美*

*山梨大学工学部コンピュータ・メディア工学科
†Computer Science Department, Virginia Tech

シナリオにもとづく設計法とは，シナリオをシステム開発工程における主要な成果物とみなし，積極的に利用する手法である．本稿では，システム設計におけるシナリオの役割に関して，代表的な研究の分析をおこなう．具体的には，代表的な４分野（戦略経営，人とコンピュータとの相互作用，要求工学，オブジェクト指向分析・設計）におけるシナリオの特徴とスコープの違いを明らかにすることで，シナリオがシステム開発に携わる多様なステークホルダに対する共通言語となり得ることを示す．

Surveying scenario-based approaches in system design

Kentaro Go*, John M. Carroll†, and Atsumi Imamiya*

*Department of Computer Science and Media Engineering, Yamanashi University
†Computer Science Department, Virginia Tech

Scenario-based design is a design methodology that uses scenario as a central artifact in system design. This article analyzes the roles of scenarios in system design based on a survey of the major scenario-based design approaches. To be precise, it examines the characteristics and differences on scopes of scenario among the four major fields: Strategic Management, Human-Computer Interaction, Requirements Engineering, and Object-Oriented Analysis and Design. Also, it shows that scenarios can provide a common language for design among various stakeholders who participate in design activities.

1. Introduction

Scenario-based system design is an approach that employs scenarios as a central representation throughout the entire system lifecycle [1], [6], [11]. 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 Strategic Management (SM), Human–Computer Interaction (HCI), Requirements Engineering (RE), and Object-Oriented Analysis and Design (OOAD). Recent researches in these communities have been progressing rapidly; nevertheless, their mutual contributions are not well made yet [11].

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.

2. 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 simply or implicitly express it.

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, a kind of scenario frequently used for envisioning user tasks in HCI [4].

3. Scenario-Based Approaches

We will examine four communities that actively use scenario-based approaches: SM, HCI, RE, and OOAD. SM forecasts the future environment of an organization and helps stakeholders plan actions. HCI aims to design usable computer systems that support their users' tasks in a safe and fluent manner. RE is about eliciting users' needs regarding computer systems, and about producing specifications; the specifications must be unambiguous, consistent, and complete. OOAD 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.

Each of the four communities has its own use of scenarios. SM lists "what-if" questions and their answers in a scenario to plan a course of actions. HCI uses scenarios to analyze a system and to envision a more usable system. RE acquires system requirements and specifies them by using scenarios. OOAD's scenario usage is to identify objects and data structures and to model a class hierarchy.

3.1 Strategic Management

The dominant idea in the SM community is that scenario planning is a process of envisioning and critiquing multiple possible futures. Kahn considers scenarios as an aid to thought in uncertainty [12]. He uses scenario (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."

One of the milestone papers in the field is Wack [17], 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. He distinguishes two distinct ways to use scenarios. On one hand, scenarios are used for direct forecasting. He claims that the scenario usage before the Royal Dutch/Shell case had been focussing on this use.

On the other hand, the current scenarios can be flexible tools for brainstorming. Wack discusses that scenario planning is an approach helping stakeholders to think about new possibilities. He emphasizes the iterative construction of scenarios, in which new scenarios are derived through the analysis of the old scenarios.

3.2 Human–Computer Interaction

HCI is another field that actively discusses what scenarios are and how to use them in system design [1], [4], [6]. HCI 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 the 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 description of workplace contexts.

Day-in-the-life scenarios are one of the most

powerful methods for envisioning the authentic computer use [14]. 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 [14]. 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 [5]; 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. Carroll and Rosson [3] proposed an iterative process of writing scenarios and of analyzing their psychological claims. They write textural narrative scenarios for envisioning future use of a system; then, they conduct claims analysis by listing upside and downside consequences of features of tools and artifacts in the scenarios. After that, examining the claims, they derive new scenarios. This iterative process makes the design of the system sharpen in response to analysis of its future.

Evaluation of systems is another typical use of scenarios in HCI. During the evaluation activities on system development, careful observations of a real work setting are necessary. The technique is frequently used in HCI 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 analyzed to reveal how the work is socially organized [8].

One of the significant views derived from HCI is that scenarios are not specifications. Carroll [1] contrasts two complementary perspectives: the perspectives clearly separate scenarios from specifications. Scenarios are concrete descriptions, focusing on particular instances, work driven, open-ended, fragmentary, informal, rough, colloquial, and envisioned outcomes. In contrast, specifications are abstract descriptions, focusing on generic types, technology driven, complete, exhaustive, formal, rigorous, and specified outcomes.

3.3 Requirements Engineering

Because the goal of RE 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 RE relatively concrete and process-oriented as a methodology.

Typical scenario-based approaches in RE include the *formal scenario analysis* by Hsia, Samuel, Gao, Kung, Toyoshima, and Chen [7], the *inquiry-based requirement analysis* by Potts, Takahashi, and Anton [15], and the *CREWS project* [16].

Hsia and associates [7] 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 stage uses scenarios in formal notation, which enable the system analysts to derive part of the system-requirement specification.

Potts and colleagues [15] 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 requirement documentation and 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."

The CREWS (Cooperative Requirements Engineering With Scenarios) project is a large European project on scenario use in RE [16]. 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. In the project, Rolland, Achour, Cauvet, Ralyte, Sutcliffe, Maiden, Jarke, Haumer, Pohl, Dubois, and Heymans [16] 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.

3.4 Object-Oriented Analysis and Design

OOAD 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* [10], Wirfs-Brock's *responsibility-driven approach* [18], and Koskimies, Systa, Tuomi, and Mannisto's *automated modeling support approach* [13].

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" [9]. Then, Jacobson 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.

Another object-oriented approach relating to scenarios is the responsibility-driven approach by Wirfs-Brock [18]. 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.

Koskimies and his colleagues proposed automated support for modeling object-oriented software [13]. They follow the definition of scenarios in the Object Modeling Technique (OMT), a commonly used OOAD 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' intention is to formalize scenarios in order to provide an automated support of scenario-based system development; therefore, they formalize them as event trace diagrams.

4. Relationship of the Four Communities

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

4.1 Structure of the Four Communities

The four communities—HCI, SM, RE, and OOAD—have a nested structure, in which the latter two communities can be categorized into Software Engineering (SE). This structure is illustrated as Figure 1. From inside to outside, there are three layers: SE, HCI, and SM. HCI incorporates work-oriented approaches focusing on

users' tasks and user participation.

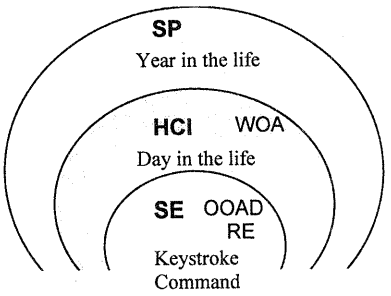

Figure 1. The nested structure of Strategic Planning (SP), Human–Computer Interaction (HCI), and Software Engineering (SE). HCI includes Work-Oriented Approach (WOA), and SE consists of Object-Oriented Analysis and Design (OOAD) and Requirements Engineering (RE).

The nested structure describes on what each field focuses based on the degree of tangibility of the target content of scenarios. Software engineering scenarios focus on real world objects or physical artifacts; also, 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, HCI scenarios treat user tasks. User tasks are not easily touched. That is, its degree of tangibility is much lower than software engineering. SM deals with much more abstracted artifacts: 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. HCI scenarios are bigger in scope; they deal with day-in-the-life. SM 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 1 also summarizes the history of HCI. HCI as it originated in software psychology studied human aspects in traditional system design approach and use of systems constructed from the approach [2]. It grew out of software engineering research and has focussed on practice. More recently, its attention has shifted to work-oriented approaches; the importance of the analysis of authentic user tasks with a system has been emphasized. Its research areas had been rapidly expanding, and researchers recognized that real work is social work though the conventional HCI research and practice 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 of intersection with SM.

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.

4.2 Lifecycle of Scenarios

The four communities make different use of scenarios in a system lifecycle; indeed, they assume a different lifecycle generally. SM and HCI presume development processes as complex, dynamic processes; therefore, they prefer to employ iterative design of scenarios. In SM, Wack described a cyclic process of scenario development [17]. Similarly, in HCI, the dominant view is that of an iterative process of design. The RE community generally assumes that development processes are decomposable; thus, techniques for the derivation of specifications from scenarios at the end of the requirement acquisition phase become a fundamental issue. The OOAD community tends to apply the object-oriented manner to any phases 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 HCI and RE.

5. Concluding Remarks: Scenario as a Common Language

Scenario-based techniques contextualize and concretize design thinking about people and technologies. SM scenarios capture organizational context; surprise-free continuations and the status quo, as well as alternative, what-if scenarios. RE scenarios help elicit and refine functional descriptions of future systems. HCI

scenarios help users and developers describe and evaluate technology currently in use, and envision activities that new technology could enable. OOAD 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 uses of scenarios emphasize different viewpoints and different resolutions of detail, and they address different purposes. But the vocabulary of concrete narrative is accessible to and sharable by diverse stakeholders in a design project: planners and managers, requirements engineers, software developers, customer representatives, HCI designers, and the users themselves. In this sense, scenarios provide a common language for design.

References

[1] Carroll, J. M. Ed. *Scenario-Based Design: Envisioning Work and Technology in System Development.* John Wiley & Sons, New York, NY, 1995.

[2] Carroll, J. M. Human–computer interaction: Psychology as a science of design. *International Journal of Human–Computer Studies, 46* (1997), 501–522.

[3] Carroll, J. M. and Rosson, M. B. Getting around the task–artifact framework: How to make claims and design by scenario. *ACM Transactions on Information Systems 10*, 2 (1992), 181–212.

[4] Chin, G., Rosson, M. B., and Carroll, J. M. Participatory analysis: Shared development of requirements from scenarios. In *Proceedings of CHI 97 Conference* (March 22-27, Atlanta, GA). ACM, New York, 1997, pp. 162–169.

[5] Dubberly, H. and Mitch, D. *The Knowledge Navigator.* Apple Computer, videotape, 1987, appears in B. A. Myers ed. HCI'92 Special Video Program.

[6] Go, K., Carroll, J. M., and Imamiya, A. Bringing user's view to design: Roles of scenario in system design, *IPSJ Magazine 41*, 1 (2000, to be published).

[7] Hsia, P., Samuel, J., Gao, J., Kung, D., Toyoshima, Y., and Chen, C. Formal approach to scenario analysis. *IEEE Software 11*, 3 (1994), 33–41.

[8] Hughes, J. A., Randall, D., and Shapiro, D. Faltering from ethnography to design. In *Proceedings of CSCW'92 Conference* (Oct. 31–Nov. 4, Toronto, Canada). ACM, New York, 1992, pp. 115–122.

[9] Jacobson, I. The use-case construct in object-oriented software engineering. In J. M. Carroll (Ed.), *Scenario-Based Design: Envisioning Work and Technology in System Development* (1995), 309–336. John Wiley & Sons, New York, NY

[10] Jacobson, I., Christersson, M., Jonsson, P., and Overgaard, G. *Object-Oriented Software Engineering: A Use-Case Driven Approach.* Addison-Wesley, Reading, MA, 1992.

[11] Jarke, M., Bui, X. T., and Carroll, J. Scenario management: An interdisciplinary approach, *Requirements Engineering 3*, 3-4 (1998), 155–173.

[12] Kahn, H. *Thinking about the unthinkable.* Horizon Press, New York, 1962.

[13] Koskimies, K. Systa, T., Tuomi, J., and Mannisto, T. Automated support for modeling OO software. *IEEE Software 15*, 1 (1998), 87–94.

[14] Mountford, S. J. A day in the life of ... (Panel). In *Proceedings of CHI'91* (April 27–May 2, New Orleans, LA). ACM, New York, 1991, pp. 385–388.

[15] Potts, C., Takahashi, K., and Anton, A. I. Inquiry-based requirements analysis. *IEEE Software 11*, 2 (1994), 21–32.

[16] Rolland, C., Achour, C. B., Cauvet, C., Ralyte, J., Sutcliffe, A., Maiden, N. A. M., Jarke, M., Haumer, P., Pohl, K., Dubois, E., and Heymans, P. A proposal for a scenario classification framework. *Requirements Engineering 3*, 1 (1996), 23–47.

[17] Wack, P. Scenarios: Uncharted waters ahead. *Harvard Business Review 63*, 5 (1985), 72–89.

[18] Wirfs-Brock, R., Wilkerson, B., and Wiener, L. *Designing Object-Oriented Software.* Prentice Hall, Englewood Cliffs, NJ, 1990.