

ScenIC: A Strategy for Inquiry-Driven Requirements Determination

Colin Potts

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280 USA

+1 404 894-5551

potts@cc.gatech.edu

ABSTRACT

ScenIC is a requirements engineering method for evolving systems. Derived from the Inquiry Cycle model of requirements refinement, it uses goal refinement and scenario analysis as its primary methodological strategies. ScenIC rests on an analogy with human memory: semantic memory consists of generalizations about system properties; episodic memory consists of specific episodes and scenarios; and working memory consists of reminders about incomplete refinements. Method-specific reminders and resolution guidelines are activated by the state of episodic or semantic memory. The paper presents a summary of the ScenIC strategy and guidelines.

Keywords

Requirements analysis, System evolution, Scenarios

1 INTRODUCTION

A project's requirements documentation or models is its memory. Psychologists have long studied memory, categorizing it along functional and temporal dimensions. A persistent theme has been the distinction between working and long-term memories. Human memory suggests useful parallels with software development, because the functions of the corresponding memory systems are analogous. Human working memories are used during a task and are then forgotten, and so project working memories consist of pending issues and decisions. Within long-term human memories, a distinction is also made between semantic memories, which capture vocabulary, generalizations and principles, and episodic memories, which recollect events, situations and cases [3]. Project semantic memories, therefore, are rules and generalizations

that apply to all use situations. In ScenIC, these are goal, task and actor specifications. Project episodic memories are specific behaviors, traces, cases, and scenarios that seem noteworthy.

Previous studies of projects have shown that project-level distraction leads to project forgetfulness and the disruption of ongoing tasks [4, 5]. A methodology guides the allocation of attention to requirements and design issues. ScenIC is designed with this attentional analogy in mind.

ScenIC instantiates the Inquiry Cycle [1, 2], the cyclical application of the following steps: expression of semantic or episodic ideas, raising and resolution of issues (criticism), and refinement of long-term memory (see Figure 1). Expression is supported by adopting semantic and episodic memory schemas, criticism by issue-raising guidelines that direct attention. Refinement is supported by resolution guidelines that suggest refinements. This model of mediated issue-raising and resolution derives from earlier research into design rationale [6, 7].

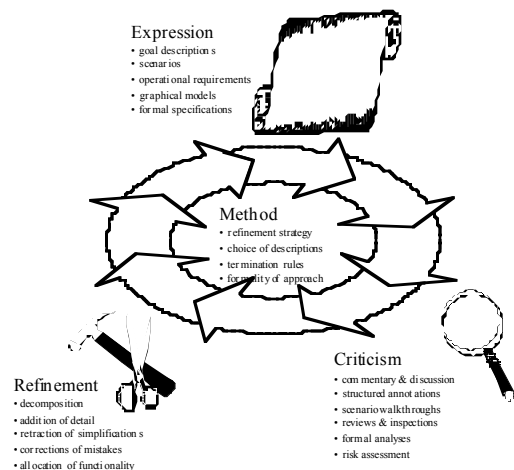


Figure 1: The Inquiry Cycle

2 THE SCENIC MEMORY SCHEMAS

Each of the three ScenIC memories has an entity-relationship schema. Detailed specification of schema elements is left undefined so that ScenIC may be applied at varying levels of formality.

Semantic Memory Schema

ScenIC semantic memory reflects information about the system. Because semantic memory changes during refinement, the “system” may include both “environment” and “machine” [8]. Semantic entities in ScenIC include actors, goals and obstacles (see Figure 2).

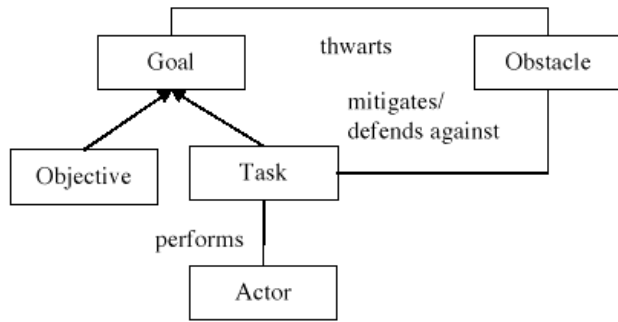


Figure 2: ScenIC semantic model schema in UML notation.

Actors are entities that participate in changes of state. Examples of external actors include user roles, organizations, physical devices, external systems and Nature. Internal actors are putative architectural components, or the whole system viewed as a black box.

There are two kinds of *goals*: objectives and tasks. *Objectives* are expressed by a trajectory of improvement (e.g. “reduce customer complaints”) or the preservation or prevention of states of affairs (e.g. “the elevator should never move with its doors open”).

Tasks are goals that are stated in terms of achievement of a state or performance of an action. (E.g. once a meeting request is entered into the system, a time and place will eventually be scheduled.)

Objectives cannot be directly achieved by a machine, but are indirectly realized through combinations of tasks. For example, to reduce customer complaints, users and machine perform detailed coordinated actions. Tasks, in contrast, may be realized directly by actors, although some may require further expansion. The specification of the system is the set of specifications of tasks allocated to internal actors.

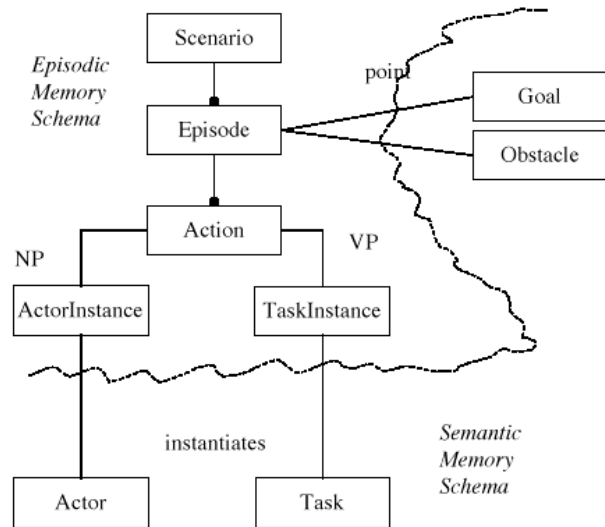
Goals may be thwarted by *obstacles*. Many are inherent in the goals themselves (e.g. unsatisfiability of a constraint), but others arise because of failure modes of the actors to which tasks are allocated. Obstacles include non-normative behaviors in general, not just failures. Non-mechanical actors may behave in unanticipated ways that violate assumptions required for goal achievement. Frequent or severe obstacles must be resolved either by *defensive* or

mitigation tasks.

ScenIC Episodic Memory Schema

User and customer representatives cannot always generate and envisage consequences from specifications, so a specified system may seem more desirable than its implementation. These arguments have been voiced to justify prototypes [9], executable specifications and models, mockups and storyboards [10, 11], walkthroughs of work-practice data [12-14], and scenarios [1, 15-22].

ScenIC episodic memory (see Figure 3) consists of episodes, sequences of purposeful task instances. Although scenarios present a compelling view of system properties, they are not necessarily representative. Their specificity means that they cannot cover the space of possible



behaviors and so are chosen to maximize salience [23].

Figure 3: ScenIC episodic memory schema.

A ScenIC episodic memory consists of goal-directed and obstacle-illustrating episodes bundled as scenarios at varying levels of detail. Scenarios are used to explore possibilities such as alternative allocations of tasks to actors, alternative obstacle resolution methods, and alternative future missions. The narrative structure of scenarios is a simplified story grammar [24-26].

ScenIC Working Memory Schema

Working memory is action-oriented. Its contents remind and direct inquiry. A project working memory in ScenIC is a record of pending issues or refinement decisions not yet acted on (see Figure 4). The working memory schema could be extended to IBIS or more complex schemas [6, 7, 27-29], and working memory could be captured or archived as rationale [27, 30]. Methodological guidance is provided by posing standard issues about long-term information. ScenIC guidelines are encoded as reminder templates with triggering contexts.

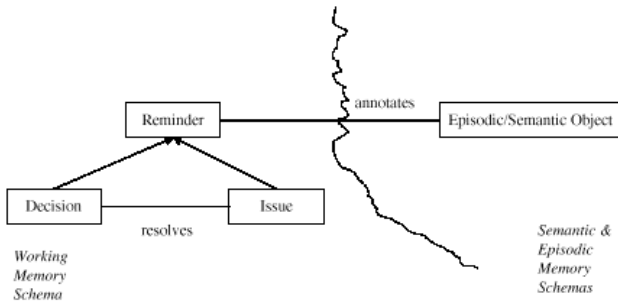


Figure 4: Working memory schema.

3 SCENIC STRATEGY AND GUIDELINES

ScenIC instantiates the expression phase of the Inquiry Cycle by providing episodic and semantic memory schemas for information about the proposed system. Criticism is supported by standard issues that are triggered by the state of the long-term memories. Refinement is supported by resolution guidelines. The framework is outlined in Table 1.

Table 1: ScenIC strategy and guidelines framework.

Triggering condition	Issue	Resolution guideline	Resulting change
Condition of semantic or episodic memory.	Issue to raise about system requirements.	ScenIC suggestions.	Condition of semantic or episodic memory.
E.g. actor performing task in an episode.	E.g. Can actor fail?	E.g. Analyze risk of failure.	E.g. New obstacle and scenarios illustrating it.

Scoping the System and Identifying Actors

External actors exist outside the designers' control. An external actor is included if it performs significant actions within the scope of the system (see Table 2).

Example: the external and internal actors in a meeting scheduling system may be shown in the following context diagram (Figure 5), extended beyond the rules of Structured Analysis [31-33] so that (1) Interactions among external actors are noted when the system/environment boundary is not fixed and an understanding of the functioning of the system requires an understanding of them; (2) Interactions are in the form of dependencies, not data flows; (3) Actors may be specialized or sub-classed to simplify interaction descriptions; (4) Actors may be classified (person, system, flight vehicle, natural environment, etc.); (5) The black box is made "translucent" to show components responsible for system functions.

Table 2: Action identification guidelines

Triggering condition	Issue	Resolution guideline	Resulting change
No identified external actors	What are the external actors?	(1) Either (a) actor performs actions or has responsibilities relevant to system's purpose, or (b) Actor interacts directly with it (2) Look for people, teams, organizations, devices and systems, and elements of environment.	Extended context diagram.
No identified internal actors.	What are the internal actors?	If internal architecture is unknown or unconstrained, do not decompose further.	Black-box, single-actor system.
Existing architecture	What are the internal actors?	(1) Include only components that correspond to physical components. (2) Components interact directly with environment or with other components that do. (3) Components exist in the architecture, are new in the new version, or result from reorganization of current architecture to which design team is committed.	Gray-box, multi-actor system.

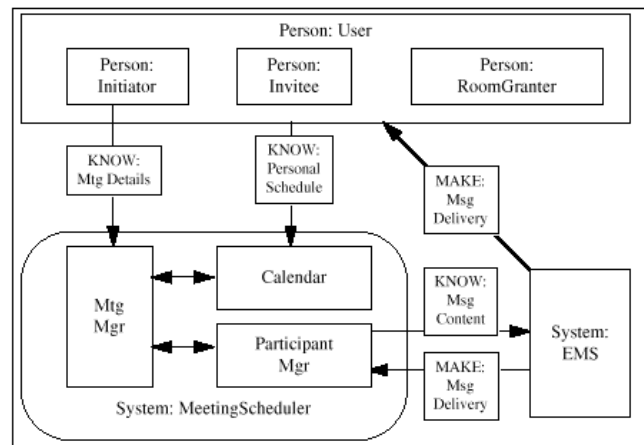


Figure 5: "Gray-box" architecture, showing external and internal actors and goal dependencies between them.

Goal Identification and Refinement

Goals are identified by reflecting on the system's purpose, interviewing stakeholders, or inferring goals from background documentation. Anton's dissertation [34]

contains suggestions on identifying enterprise goals. Goals may also be identified from long-term memory content. Scenarios, for example, may illuminate previously ignored or underemphasized goals. Obstacles may suggest subsidiary tasks. Table 3 lists ScenIC guidelines for identifying goals.

Table 3: Goal identification guidelines

Triggering condition	Issue	Resolution Guideline	Resulting change
No goals	What are the goals for the system?	Obtain from mission statements, questions to stakeholders, etc.	Semantic memory with goals.
Episode with behavior but no or partial point.	Why is this actor doing that?	(Default) Actions often define tasks one-to-one. Ask “why?” to identify neglected objectives.	(1) Episode with point. (2) Task ascribed to actor. (3) (possibly) higher-level objectives.
An obstacle that must be defended against or mitigated but which is not.	How should the obstacle be defended against? How should it be mitigated if it occurs?	Standard preventive maintenance and risk analysis methods.	Maintenance objective: “Avoid <obstacle>” and its expansion into tasks. Achievement goal that encapsulates mitigation strategy.

Table 4: Types of goal and suggested verbs for labeling.

Object of goal achievement	Plausible verbs
Improving condition	Improve, reduce, maximize...
Maintaining condition	Keep, maintain, preserve
Avoiding condition	Avoid, keep from, prevent...
Enabling actor to achieve a goal (or preventing)	Enable, let, empower, give, disable, prevent...
Satisfying condition	Satisfy, achieve
Bringing about state	Make, create, achieve...
Bringing about knowledge	Know, find out, ascertain,...
Bringing about commitment	Decide, select, agree,...
Providing knowledge	Tell, inform, notify, remind...
Soliciting knowledge	Ask, request...
Providing knowledge about	Identify, distinguish, detect...

A lexicon of verbs is useful for identifying objectives and tasks (Table 4). Example objectives include “Maximize utilization of meeting rooms” and “Avoid boring meetings.” Example tasks include “Satisfy meeting requests” and “Know availability of rooms.”

Goals are expanded by means-end analysis (see Table 5). There are generic guidelines for expanding maintenance objectives and tasks, but improvement objectives tend to be domain-specific.

Table 5: Refinement of goals into expanded subgoals.

Triggering condition	Issue	Resolution Guideline	Resulting change
MAINTAIN p.	What are the conditions of p?	Identify properties (p ₁ ..p _n) that contribute to p.	MAINTAIN p ₁ & ... MAINTAIN p _n
AVOID p (or MAINTAIN ~p)	What are the indicators of p?	Identify early-warning signs (q) of p and any counteracting tasks (anti-p).	DETECT q & MAKE anti-p
MAKE p	How to satisfy p?	Identify & combine knowledge [q] and action [r] sub-tasks.	KNOW q & MAKE r
KNOW k	How to find out k?	(1) Ask another actor (2) Perform background computation	MAKE (a, asked[k]) & MAKE (a, asked[d, k']) KNOW k ₁ & ... KNOW k _n

Task Dependencies

Tasks often must be done in order, because they depend on initiating conditions. These temporal orderings are implicit in inter-task dependencies. High-level tasks gradually consume information or depend on partial completion of dependent tasks. For this reason, it is best to defer analysis of task dependencies until tasks are detailed and could plausibly be allocated to actors. Sometimes a task depends on the availability of information without it being obvious how the information is obtained. An additional knowledge goal is then needed (see Table 6).

Table 6: Task dependency guidelines

Trigger	Issue	Guideline	Change
Task t= MAKE p	What are t’s usual preconditions?	What must be true for p to be true?	t <u>requires</u> r (where a task t ₂ =MAKE r)
Task t	What information does t need?	Information t or any knowledge sub-tasks use.	(KNOW k & t _i ...) <u>expand</u> t

Allocation of Tasks to Actors

Which tasks should be done by the system and which by external actors is seldom pre-ordained. A system could decide the best time and place for a meeting automatically. Alternatively, it could collate information and present it to the user for a final decision. Both systems support meeting scheduling goals but have different boundaries.

Task allocation is aided by scenario analysis. Alternative allocations can be discussed in alternative scenarios, and the issues raised used in evaluating them. Possible allocations are determined in part by the current architecture and in part by current practices (see Table 7).

Table 7: Task allocation guidelines

Triggering conditions	Issue	Resolution guideline	Resulting change
Actors and unallocated tasks	What allocation is possible?	Constraints afforded by architecture and practices.	Candidate allocations
Actors and unallocated tasks	What task allocation is feasible and economic?	Domain-specific. Architectural analysis with scenarios [40].	Candidate allocations
Candidate allocation	Can actor fail?	Failure modes for actor type.	Induced obstacles
Objectives and candidate allocation	How does allocation affect objectives?	Episode-by-episode scenario analysis	Weakened objectives or revised allocation

Obstacle Identification

Obstacles can be identified by a number of processes that range from the most informal to systematic engineering practices. Generally, obstacles are approached top-down and bottom-up (see Table 8). Table 9 categorizes some frequently occurring types of obstacles.

(1) *Top-down identification of obstacle combinations that block objectives* Single obstacles are seldom responsible for an objective not being achieved. For example, “Avoid boring meetings” can be thwarted by a combination of factors. If this is a legitimate goal for the system, it is necessary to ask how people characterize meetings as boring, how they avoid them, and how a system might make it difficult to do so. A minimal top-down strategy is generate-and-test: systematically asking whether each allocation decision undermines an objective. Another is to construct anti-scenarios from critical incidents.

(2) *Bottom-up identification of obstacles that block tasks.* Some obstacles are inherent in the problem domain. For example, a best place for a meeting cannot be chosen if there are no rooms available. On the other hand, many arise from failure modes of actors. For example, obstacles that block finding out when people are available arise from human failings or system failures: people may not update

their online calendars, or a calendar may get corrupted.

Table 8: Obstacle identification guidelines

Triggering conditions	Issue	Resolution guideline	Resulting change
Objectives and possibly allocations	What obstacles can block the satisfaction of this objective?	Generate episodes and allocations and test against objective Consider anti-scenarios or known negative cases	Weakened objective and/or revised episodes and allocations Obstacles
Goals and possibly allocations	What could cause this goal to be blocked?	Consider inherent obstacle types (e.g. resource unavailability) Consider failure modes specific to actor type	Obstacles

Table 9: Categories of obstacle with examples.

Categories of Obstacle	Example obstacles
Actor failure	Calendar database unavailable.
Medium failure	Email message does not arrive.
Resource contention	There are no rooms available at that time.
Replicate confusion	Wrong John Smith invited to meeting.
Inherent goal conflicts	Taxpayer convenience is thwarted by need to audit returns.

Elaboration of Objectives and Tasks

There are three ways to deal with an obstacle: (1) ignore it; (2) defend against it, ideally preventing it altogether; and (3) mitigate its effects, ideally recovering altogether. In many cases, defense is more effective and less expensive than mitigation, but when the obstacle’s consequences are mild and the cost of defense is high, it makes sense to rely on mitigation. Mitigation is only worth considering when defense is likely to fail and the risk of the residual obstacle is still worth worrying about. In Rational Management [36], tables like Table 10 are used to list defensive and mitigation strategies. Similarly, RPM [35], a systems engineering method for preventive maintenance, translates directly into such issues (see Table 11).

Table 10: Defense and mitigation in Rational Management.

Obstacle	Prob.	Defensive strategy	Residual prob.	Mitigation strategy
Unavailability of room at good time	High	Quorum-based reservation.	Medium	Relax invitees’ schedules

Table 11: RPM guidelines for defense and mitigation

Trigger	Issue	Guideline	Change
Hidden obstacle	Can hidden obstacle be defended against?	Likelihood of defensive strategy also failing is acceptably low.	Early-warning and remediation strategy
Obstacle	Hazardous?	Defensive strategy must reduce obstacle likelihood to acceptable level.	Early-warning and remediation strategy, with logging
Non-hazardous obstacle	Should it be defended against?	Cost should outweigh cost of defense.	Periodic check / early-warning

4 THE INQUIRY CYCLE AND SCENARIOS

The point of exploring scenarios is to raise and resolve issues about requirements. Many types of issues may be relevant: What else can happen? Is this obstacle worth worrying about? How frequent/severe is it? How else could we prevent this from happening or mitigate its effects? One benefit of scenario analysis is the serendipitous insights that a scenario may yield. When an issue that was raised from a scenario is resolved, the resulting changes may occur anywhere in episodic or semantic memory.

Episodes follow directly from task structure. Scenarios are composed from episodes so that normal cases and anti-scenarios are elaborated. The guidelines for identifying and elaborating episodes are summarized in Table 12 and those for identifying and composing scenarios in Table 13.

Having identified obstacles, it is possible to compose any number of scenarios that contain normative (goal satisfaction) and exceptional (obstacle occurrence) episodes, most of which are redundant or uninteresting. We would like to put our early effort into the most salient scenarios, the ones will tell us the most [23]. Scenario composition is therefore a form of sampling, like test-case generation. But scenarios differ from test cases in two important respects. First, scenarios are used to discover and elaborate requirements, so any measure of scenario salience or coverage is a moving target. Secondly, we can assess the adequacy of test cases against an objective baseline, whereas any requirements baseline is subjective. The ScenIC solution is pragmatic: The set of goals and obstacles is taken as the baseline for assessing scenario coverage. As the analysis of scenarios helps in the identification of further goals and objectives, so the heuristics shown in Table 14 should be reapplied, leading to the revision of existing scenarios, and possibly the composition of new ones.

(1) *Normal-case coverage.* There should be at least one scenario for each normal case. There are usually alternative normal cases, some of which cover initiation or system-maintenance goals.

(2) *Single-obstacle coverage.* Each obstacle should occur in at least one scenario.

Table 12: Episode identification guidelines

Trigger	Issue	Guideline	Change
MAKE p.	What episodes make p?		EPISODE: making p
POINT a makes p	How is p accomplished?	Goal expansion.	[TASK (b, p1) & ...TASK(c, pn)] expand TASK(a, p) EPISODE: making p: b p1 ; ... c pn
EPISODE: making p & other episodes	How are preconditions established?	Identify tacit preconditions. & episodes to establish.	TASK p requires q & MAKE q & EPISODE making q: POINT: make q

Table 13: Guidelines for identifying scenarios.

Trigger	Issue	Guideline	Change
Episodes and goals	Main-line use case?	Orient scenario episodes around major tasks.	New scenarios / episodes.
Episodes and goals	Initiating / maintenance scenarios?	(1) Scenarios involving administrative actors. (2) Tasks establishing preconditions.	New scenarios / episodes
Episodes, goals and obstacles	Salient, single-case exception scenarios?	Significant obstacles for each task in normative case.	New scenarios and episodes
Episodes, goals and obstacles	Salient obstacle-combinations?	Significant obstacle co-occurrences. Risk analysis.	If risk, obstacle expansions
Objectives	Tasks / obstacles that contribute or detract?	Elaborate scenario to illustrate achievement of objective	Scenario or anti-scenario

Figure 14: Guidelines for identifying salient scenarios.

Trigger	Issue	Guideline	Change
Goals, obstacles, episodes	How many and what scenarios?	Normal cases & obstacle coverage.	Salient scenarios
Critical incidents	How many and what scenarios?	Critical incidents as anti-scenarios	Salient anti-scenarios

(3) *Coverage of objectives.* Ideally, every scenario should be assessed against every objective. For greater efficiency, it is advisable to divide the objectives into two: those that absolutely must be achieved, and those that are strong preferences but could be violated. Objectives of the first

type should be checked when elaborating every scenario, whereas those of the second type may be sampled randomly.

5 DISCUSSION

The Inquiry Cycle is a general approach to improvement of documented ideas. In ScenIC, these ideas are about enhancements to an existing system, so ScenIC interacts with other techniques for system evolution from the MORALE program [38], including methods for reverse engineering [39] and architectural evaluation [40].

However, the principle of using directed inquiry as a working memory applies to codifying any method for producing, organizing, refining or evaluating knowledge work. Previous studies have explored and evaluated the application of the Inquiry Cycle [2] and GBRAM [37] to industrial projects. ScenIC has been or is currently being applied to the domains of web browsing and telephone call processing management.

We have deliberately de-emphasized language issues, because the three ScenIC memories could be implemented at varying levels of formality without affecting the principles of project attention management. For example, goals could be represented as text statements, graphical goal networks [51] or logical formulae [48]. Scenarios could be represented as narrative texts, structured or tabular scripts [20] or formal traces [41]. And reminders could consist of text annotations, semi-structured hypertext networks [6, 27, 29], or formal change dependencies.

ScenIC emphasizes scenarios, in which respect it resembles other efforts in requirements engineering [15, 16, 18, 19, 21, 23, 40, 41], object-oriented design [20, 22, 42], usability engineering [10, 11, 17, 23], and strategic planning [43-46]. Previous work has addressed the integration of scenarios into the Inquiry Cycle [1, 2]. ScenIC resembles other requirements-engineering approaches that start with goals [1, 16, 34, 37, 47-51]. It also incorporates insights and techniques from artificial intelligence [52], business planning [53], decision theory [54], human-computer interaction [55], management science [36], maintenance engineering [35], soft-systems methodology [56], and software metrics [57]. Obstacle identification, defense and mitigation owe more to project planning, industrial engineering and manufacturing than to software engineering. The methods of fault-tree analysis (FTA) and failure-modes effects analysis (FMEA) are described in numerous books on safety and reliability engineering [58]. There is a good tutorial summary of contingency planning in Rational Management [36].

Thinking of human short-term memory as a working memory stems from work by Baddeley [59], and the distinction between semantic and episodic memories from Tulving [3]. The ScenIC scenario schema derives from the story grammars of Thorndyke [26] and Rumelhart [25].

Design rationale, or the retention of organizational memories about why design decisions were made has been

discussed for many years (see [60]). There have been promising accounts of the value of rationale in large projects [30], but the notion of rationale “capture” fails to recognize important differences between the requirements on long-term and short-term memories. Short-term memories are superficially coded and organized, but are easy to find and are the focus of attention; long-term memories, in contrast, are richly organized. Early investigators ignored the cost of reorganizing and integrating rationale so that it could be retrieved meaningfully later, a cost usually not borne by those who would benefit [61].

We are developing tool support for expression and criticism in ScenIC. This is based on the use of hypermedia annotations demonstrated in Ecolabor [62].

ACKNOWLEDGEMENTS

Effort sponsored by the Defense Advanced Research Projects Agency and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-96-2-0229. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, Rome Laboratory, or the US Government.

REFERENCES

1. Potts, C., K. Takahashi, & A.I. Antón, Inquiry-Based Requirements Analysis. *Software*, 1994. 11(2): 21-32.
2. Potts, C., et al. An Evaluation of Inquiry-Based Requirements Analysis for an Internet Service. in *IEEE Symp. Reqs. Eng. (RE'95)*. 1995. York, UK.
3. Tulving, E., Episodic and Semantic Memory, in *Organization of Memory*, E. Tulving, (Ed.). 1972, Academic Press: New York.
4. Catledge, L. & C. Potts. Collaboration During Conceptual Design. In *Int. Conf. Reqs. Eng. (ICRE '96)*. 1996. Colorado Springs, CO: IEEE CS.
5. Potts, C. & L. Catledge, Collaborative Conceptual Design: A Large Software Project Case Study. *Computer-Supported Cooperative Work*. 1996. 5: 414-445.
6. Potts, C. & G. Bruns. Recording the Reasons for Design Decisions. in *10th Int. Conf. Software Eng.*. 1988. Singapore: IEEE CS Press.
7. Potts, C. A Generic Model for Representing Design Methods. in *11th Int. Conf. Software Eng.*. 1989. Pittsburgh, PA: IEEE CS Press.
8. Zave, P. & M. Jackson, Four dark corners of requirements engineering. *ACM Trans. Software Eng.*, 1997. 6(1):1-30.
9. Vonk, R., *Prototyping: The effective use of CASE technology*. 1990, Prentice Hall.
10. Muller, M., et al., Bifocal tools for scenarios and representations in participatory activities with users, in *Scenario-Based Design: Envisioning Work and Technology in System Development*, J.M. Carroll, (Ed.). 1995, Wiley.
11. Ehn, P., The Art and Science of Designing Computer Artifacts, in *Scand. J. Inf. Sys.* 1989 21-42.
12. Beyer, H.R. & K. Holtzblatt, Apprenticing with the customer. *Comm. ACM*, 1995. 38(5): 45-54.
13. Brun-Cottan, F. & P. Wall, Using video to re-present the user. *Comm. ACM*, 1995. 38(5): 61-70.
14. Holtzblatt, K. & H. Beyer, Making customer-centered

- design work for teams. *Comm. ACM*, 1993. 36(10): 92-103.
15. Anderson, J.S. & B. Durney. Using Scenarios in Deficiency-driven Requirements Engineering. in *Int. Symp. on Reqs. Eng. (RE'93)*. 1993. San Diego, CA: IEEE CS.
 16. Antón, A.I., W.M. McCracken, & C. Potts. Goal Decomposition and Scenarios Analysis in Business Process Reengineering. in *Adv. Inf. Sys. Eng.: 6th Int. Conf., CAiSE '94*. 1994. Utrecht, Netherlands: Springer.
 17. Carroll, J.M. & M. Rosson, Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario. *ACM Trans. Inf. Sys.*, 1992. 10(2): 181-212..
 18. Filippidou, D., Designing with Scenarios: A Critical Review of Current Research and Practice. *Reqs. Eng.*, 1998. 3(1): 1-22.
 19. Rolland, C., et al., A Proposal for a Scenario Classification Framework. *Reqs. Eng.*, 1998. 3(1): 23-47.
 20. Rubin, K.S. and A. Goldberg, Object behavior analysis. *Comm. ACM*, 1992. 35(9): 48-62.
 21. Sutcliffe, A., Scenario-Based Requirements Analysis. *Reqs. Eng.*, 1998. 3(1): 48-65.
 22. Jacobson, I., *Object-Oriented Software Engineering: A Use-Case Driven Approach*. 1992: Addison-Wesley.
 23. Potts, C. Using Schematic Scenarios to Understand User Needs. in *Symp. Designing Interactive Systems*. 1995, Ann Arbor, Michigan: ACM.
 24. Bartlett, F.C., *Remembering*. 1932, Cambridge Univ. Press.
 25. Rumelhart, D., Notes on a Schema for Stories, in *Representation and Understanding*, D. Bobrow & A. Collins, (Eds.) 1975, Academic Press.
 26. Thorndyke, P.W., Cognitive Structures in Comprehension and Memory for Narrative Discourse. *Cog. Psych.*, 1977. 9: 77-110.
 27. Conklin, J. & M. Begeman, gIBIS: A Tool for all Reasons. *J. Am. Soc. Inf. Sci.*, 1989(May): 200-213.
 28. Kunz, W. & H. Rittel, *Issues as Elements of Information Systems*, 1970, Inst. Urban and Regional Development, Univ. Calif.: Berkeley, California.
 29. Lee, J.-t. Extending the Potts and Bruns Model for Recording Design Rationale. in *15th Int. Conf. on Software Eng.*. 1991: IEEE CS Press.
 30. Burgess-Yakemovic, K.C. & J. Conklin. Report on a Development Project Use of an Issue-Based Information System. in *CSCW'90*, 1990. ACM.
 31. DeMarco, T., *Structured Analysis and System Specification*. 1979, Englewood Cliffs, New Jersey: Prentice-Hall.
 32. Ross, D.T., Structured Analysis: A Language for Communicating Ideas. *IEEE Trans. Software Eng.*, 1977. 3(1): 16-34.
 33. Ross, D.T. & K.E. Schoman, Structured Analysis for Requirements Definition. *IEEE Trans. Software Eng.*, 1977. 3(1): 6-15.
 34. Antón, A.I., *Goal Identification and Refinement in the Specification of Software-Based Information Systems*, PhD Dissertation. 1997, Georgia Inst. Technol. College of Computing: Atlanta, GA.
 35. Moubray, J., *Reliability-Centered Maintenance*. 1992, Industrial Press.
 36. Kepner, C.H. & B.B. Tregoe, *The Rational Manager: A Systematic Approach to Problem Solving and Decision Making*. 2nd ed. 1976, Princeton, NJ: Kepner-Tregoe, Inc.
 37. Antón, A.I. & C. Potts. The Use of Goals to Surface Requirements for Evolving Systems. in *20th Int. Conf. Software Eng.* 1998. Kyoto, Japan: IEEE CS.
 38. Goel, A., et al. MORALE: Mission Oriented Architectural Legacy Evolution. In *Int. Conf. Software Maintenance*. 1997. Bari, Italy.
 39. Moore, M. Rule-Based Detection for Reengineering User Interfaces. In *Proc. 3rd Working Conf. Reverse Eng.* 1996. Monterey, CA: IEEE CS Press.
 40. Kazman, R., et al., Scenario-Based Analysis of Software Architecture. *Software*, 1996(November): 47-55.
 41. Hsia, P., et al., Formal Approach to Scenario Analysis. *Software*, 1994. 11(2): 33-41.
 42. Texel, P.P. & C.B. Williams, *Use Cases Combined with Booch, OMT, UML: Process and Products*. 1997: Prentice-Hall.
 43. Heijden, K.v.d., *Scenarios: The Art of Strategic Conversation*. 1996, Chichester, UK: Wiley.
 44. Ringland, G., *Scenario Planning: Managing for the Future*. 1998, Chichester: Wiley.
 45. Schwartz, P., *The Art of the Long View*. 1991: Doubleday.
 46. Waltre, M., *Scenario Analysis: An Approach to Organisational Learning*, 1996, Dissertation, Stockholm Univ./Royal Inst. Technol.: Stockholm, Sweden.
 47. Antón, A.I. Goal-Based Requirements Analysis. in *Int. Conf. Reqs. Eng. (ICRE '96)*. 1996. Colorado Springs, CO, USA: IEEE CS Press.
 48. Dardenne, A., A.V. Lamsweerde, & S. Fickas, Goal-directed requirements acquisition. *Sci. Comp. Prog.*, 1993. 20(1-2): 3-50.
 49. Green, S., *Goal-Driven Approaches to Requirements Engineering*. 1994, Tech Report, Imperial Coll. Sci. Technol. Med., Dept. Computing: London.
 50. McDermid, J., Requirements Analysis: Orthodoxy, Fundamentalism and Heresy, in *Requirements Engineering: Social and Technical Issues*, M. Jirotko & J. Goguen, (Eds.). 1994, Academic Press.
 51. Yu, E. Towards modeling and reasoning support for early phase requirements engineering. in *Proc. RE97: 3rd Int. Symp. Reqs. Eng.*. 1997. Annapolis, MD: IEEE CS Press.
 52. Schank, R.C. & R.P. Abelson, *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. 1977, Erlbaum.
 53. Rouse, W.B., *Best Laid Plans*. 1995, PTR Prentice Hall.
 54. Keeney, R.L., *Value-Focused Thinking: A Path to Creative Decisionmaking*. 1992, Harvard Univ. Press.
 55. Card, S.K., T.P. Moran, & A. Newell, *The Psychology of Human-Computer Interaction*. 1983, Erlbaum.
 56. Checkland, P. & S. Holwell, *Information, Systems and Information Systems: Making Sense of the Field*. 1998, Chichester, UK: Wiley.
 57. Basili, V. & D. Rombach. Tailoring The Software Process to Project Goals and Environments. in *9th Int. Conf. Software Eng.* 1987. Monterey, CA: IEEE CS.
 58. Storey, N., *Safety-Critical Computer Systems*. 1996, Harlow, UK: Addison- Wesley Longman.
 59. Baddeley, A.D. and G.J. Hitch, Working Memory, in *Advances in Learning and Motivation*, G. Bower, (Ed.). 1974, Academic Press: New York. 47-90.
 60. Moran, T.P. & J.M. Carroll, eds. *Design Rationale: Concepts, Techniques and Use*. 1996, Erlbaum.
 61. Grudin, J., Evaluating Opportunities for Design Capture, in *Design Rationale: Concepts, Techniques and Use*, T.P. Moran & J.M. Carroll, (eds.) 1996, Erlbaum.
 62. Takahashi, K., et al. Hypermedia Support for Collaboration in Requirements Analysis. in *2nd Int. Conf. Reqs. Eng. (ICRE'96)*. 1996. Colorado Springs, CO: IEEE CS.

