**RE 2003 Mini-tutorial**

# Scenario-based Requirements Engineering

Alistair Sutcliffe
*Centre for HCI Design, Department of Computation*
*University of Manchester Institute of Science & Technology (UMIST)*
*PO Box 88, Manchester, M60 1QD, UK*
*ags@co.umist.ac.uk*

## Abstract

This mini tutorial explains the concepts and process of scenario based requirements engineering. Definitions of scenarios are reviewed, with their informal and more formal representations, and roles in the requirements process. The relationships between scenarios, specifications and prototypes is explored, and set in the perspective of human reasoning about requirements. Methods for scenario based RE are described and one method, SCRAM, is covered in more depth. The tutorial concludes with a look forward to the future of scenario based RE and research directions.

## 1. Introduction

Scenarios have attracted considerable attention in RE, but unfortunately the term scenario has been interpreted by too many authors to have a commonly accepted meaning. The Oxford English Dictionary defines a scenario as "the outline or script of a film, with details of scenes or an imagined sequence of future events". Scenarios are used in business systems analysis as well as RE, the most common form being examples or stories grounded in real world experience.

Jarke et al. [1] reviewed approaches to scenario-based RE, research issues, and different scenario concepts; indeed their article introduces a whole issue of the *Requirements Engineering* journal devoted to use of scenarios. Rolland et al. [2] provide a good survey which distinguishes between the purpose or intended use of a scenario, the knowledge content contained within a scenario, how a scenario is represented, and how it can be changed or manipulated. Another taxonomy by Carroll [3] classifies scenarios according to their use in systems development ranging from requirements analysis (stories of current use), user-designer communication, examples to motivate design rationale, envisionment (imagined use of a future design), software design (examples of behaviour thereof), through to implementation, training and documentation. Kutti [4] distinguishes between scenarios in the wide, which describe the system and its social environment, and scenarios in the small that contain event sequences pertaining to a specific design. Anton and Potts [5] survey the different representations of scenarios in HCI, object-oriented software engineering and RE, ranging from informal narrative to formatted texts and more formal models. They also compare scenarios as models with concrete scenarios or instances that represent a single example of an event sequence [6]. Scenarios can vary from rich narrative descriptions of a system's use with information about the social environment [7] to descriptions of event sequences in tabular formats (e.g [8]) to more formal models of system behaviour [9, 10]). In object-oriented design it becomes difficult to distinguish between use cases, alternative paths through use cases, and scenarios, which are just another path through a use case [11, 12, 13].

Probably the best way to understand scenarios is as a continuum from the real world descriptions and stories to models and specifications. At one end of this dimension, scenarios are examples of real world experience, expressed in natural language, pictures, or other media [14, 15]). At the specification end are scenarios which are arguably models such as use cases, threads through use cases and other event sequence descriptions [11, 16]. Within the space of scenarios are representations that vary in the formality of expression in terms of language and media on one dimension and the phenomena they refer to on the other; ranging from real world experience, to invented experience, to behavioural specifications of designed artefacts.

## 2. Scenarios as design representations

Scenarios are related to models by a process of abstraction and to prototypes by a process of design (see figure 1). Scenarios which are representations of the real

world are generalised during requirements analysis to produce models, which are familiar to practitioners in requirements engineering (e.g. i* [17]) or software engineering (e.g. UML, [18]). Other informal representations such as design rationale [19] can capture design decisions that are anchored in a scenario-based expression of a problem. Models and requirements specifications become transformed into designs and eventually implemented. During that process scenarios which represent behaviour of designed artefacts have a role to play in validation. In this case scenarios are similar to models in both format and content, although they usually illustrate possible sequences of behaviour that might be valid within the constraints of a requirements specification [9].

In an alternative development route via prototyping, scenarios function as design inspiration, and material to test the prototype design [3, 15, 20]. Scenarios can be used for reasoning about design and as test scripts in evaluation methods [21, 22]. Carroll has articulated several different roles for scenarios in the design process including as envisionment for design exploration, requirements elicitation and validation [3, 23].

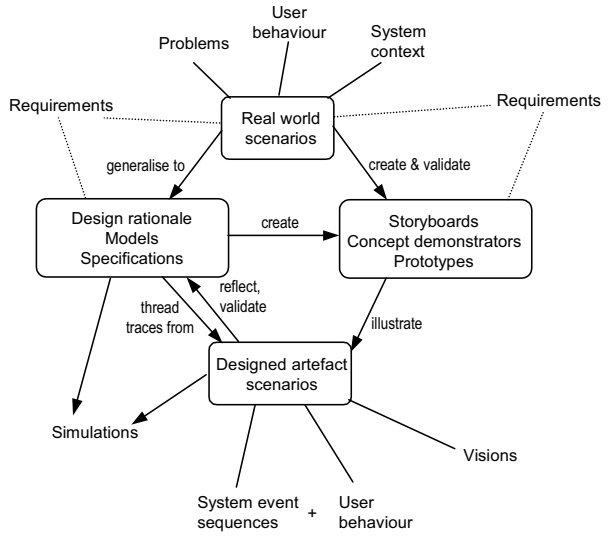


**Figure 1. Role of scenarios and their relationship to requirements specifications and prototypes.**

These views of scenarios point to three different roles:

1. A story or example of events as a grounded narrative taken from real world experience [3]. These stories are close to the common sense use of the word and may include details of the system context (scenes).
2. A future vision of a designed system with sequences of behaviour and possibly contextual description [7]. In this case the scenario comes close to a design mock up.
3. A single thread or pathway through a model (usually a use case). This is the sense in which the object-oriented community use the word [11, 13, 18]. This might be represented as an animated display of an event sequence in a message sequence or state transition diagram

Thus scenarios may vary according to their content, how closely they relate to the real world, and their role in the design process. A scenario of the following form clearly relates to the real world, is a narrative text, and expresses a user's requirements in some sense. Although it relates to the real world it is in fact made up by myself, albeit from case study experience:

> Dr Patel has been given health education targets to fulfil by the government. He hasn't time to counsel individual patients about life style and healthy living, but he is interested in how computers might help. He thinks that if a computer "advisor"-type system were placed in his surgery waiting room, patients with time on their hands might explore the system out of curiosity and learn something about healthy living. He wants to target heart disease and educate the public about the dangers of smoking, lack of exercise and other lifestyle causes of heart disease. Unfortunately most of the people who come into his surgery have seen many warnings about smoking before and seem to take no notice.

This scenario can funtion as a design brief to initiate a requirements capture process. A more detailed scenario could record a day in the life of Dr Patel and his interaction with one individual patient, Mr Hardcastle. In the latter case a scenario would be drawn from directly recorded conversation as found in ethnographic studies [24].

Depending on one's usage, scenarios are represented in different media. Scenarios that describe the real world are captured as speech or text narratives and may be embellished with photographs or videos to illustrate the context. Real world scenarios may be interpreted and then represented as formatted text and by pathways in use case or activity sequence diagrams. Designed world scenarios can be represented by anything from a storyboard sketch to animated sequences in a formal specification [25] or in a concept demonstrator [20], so scenarios start out as stories and examples but converge with models and prototypes during design.

## 3. Advantages and disadvantages of scenarios

The advantage of scenarios lies in the way they ground argument and reasoning in specific detail or examples, but the disadvantage is that being specific loses generality. We all naturally look for patterns and similarities in the world, which is the cognitive process of generalisation. In RE our reasoning process is the same. Scenarios in the real world sense are specific examples. The act of analysis and modelling is to look for patterns in real world detail, then extract the essence, thereby creating a model. So scenarios fit into the process of requirements elicitation which gathers stories and examples from users and then looks for the generalities.

Another advantage of scenarios lies in their focus on reality that forces us to address the "devil in the detail" during requirements specification and validation. In this case we confront abstract models with scenarios as test data. Whereas an abstract model can go unquestioned, unless rigorous automated reasoning is used via formal methods [26], the detail in scenarios naturally challenges assumptions in models. Examples exert a powerful effect on our reasoning because we can identify with detail through our experience. Real world scenarios are a form of episodic memory [27] that can give rich details of events and scenes, but the requirements engineer has to beware that people's episodic memory can be highly selective, so a sample of scenarios may represent atypical events from a personal viewpoint. Since scenarios situate examples with existing memory they help in understanding requirements problems. The concept of obstacles analysis in RE [28] draws on scenarios to situate our thought about how problems in the real world (i.e. obstacles) might prevent a requirement being met.

Scenarios can help to counteract pathologies in human reasoning [29], such as not testing hypotheses and assumptions in models. In RE scenarios can help to test models and specifications during requirements validation; unfortunately, scenarios can also encourage other pathologies, so we need to be on our guard. First is confirmation bias: we tend to seek only positive examples that agree with our preconceptions [30]. Scenarios should be collected which include errors, counter-examples and exceptions as well as the norm. Encysting, or not being able to see the wood for the trees, can result from becoming obsessed with the detail in scenarios. Both the model, abstract and detailed scenario views are necessary. Unfortunately the products of the generalisation process, abstract models, are not so easily understood; however, reasoning with concrete examples as well as abstract models helps comprehension by building a memory schema linking the specific (scenario) with the general (model). This can improve requirements elicitation and validation. Scenarios can bias beliefs in frequencies of events and probabilities [29], so it is important to ensure that a wide ranging sample of scenarios are gathered. This exposes one of the dilemmas in scenario based RE. Ideally the more scenarios the better to increase test coverage, but gathering and using more scenarios incurs cost. The problem is that we need many scenarios to test a general requirements specification, but how can we be sure we have a complete or even an adequate set? This can be summarised as the 20/20 foresight problem: how to capture (or generate) a sufficient set of scenarios to cover all the important problems in the application. I will return to this in section 7.

## 4. Scenarios in the requirements and design process

One particular role of scenariosis to act as a "cognitive prosthesis" or an example to stimulate the designer's imagination. Scenarios can guide thought and support reasoning in the design process [15, 31]. However, this approach can lead to errors. A scenario, or even a set of scenarios, does not explicitly guide a designer towards a correct model of the required system. An extreme scenario might bias reasoning towards exceptional and rare events, or towards the viewpoint of an unrepresentative stakeholder. These biases are an acknowledged weakness of scenarios; however, we can trust designers as knowledgeable, responsible people who are capable of recognising such biases and dealing with them productively. Indeed, some propose scenarios that are deliberately exceptional to provoke constructive thought [32]. Although scenarios are useful as cognitive probes for design, this is not their only role.

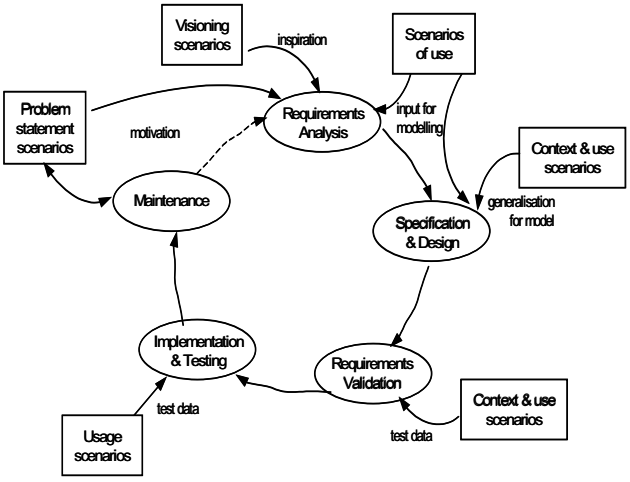


**Figure 2. Roles of scenarios in requirements and design.**

Scenarios are arguably the starting point for all modelling and design, and contribute to several parts of the design process (see figure 2). Scenarios of use describe system operations at different stages of development. Context scenarios add information about

the system's physical and social environment. At the initiation of development, scenarios play three roles: first as descriptions of the unsatisfactory state of affairs with a current system which the new system has to solve; secondly as visions of how the new system might operate; and thirdly as descriptions of behaviour, representing both users and the existing system. Usage or behavioural scenarios are a common form and can be used later in the life cycle as test data to validate requirements specifications, designs and implemented systems. Information on the system's physical and social context can be added to usage scenarios, to provide a richer input to requirements specification and validation, for instance allowing reasoning about obstacles in the system enivornment which may prevent requirements being achieved [27].

Narrative descriptions of the real world provide input to the process of generalisation that produces specific action sequences (i.e. formatted scenarios) and then a general model that represents typical behaviour of a group of users interacting with a system. The process of generalisation inevitably loses detail and the analyst has to make judgements about when unusual or exceptional behaviours are omitted, or explicitly incorporate them as alternative paths in use cases or in action sequences [11]. Indeed there is merit is eliciting or creating mis-use cases that describe threats and exception conditions that will test the system [12]. A criticism of model approaches to requirements engineering is that they inevitably omit detail which may be vital, whereas scenarios might be able to gather such detail but at the price of effort in capturing and analysing a "necessary and sufficient" set of scenarios.

## 5. Methods for scenario-based requirements engineering

One productive juxtaposition of scenarios and models is to use scenarios as test data to validate design models. This approach, proposed in the Inquiry Cycle [8] and its scuccessor ScenIC, uses scenarios as specific contexts to test the utility and acceptability of system output. By questioning the relevance of system output for a set of stakeholders and their tasks described in a scenario, the analyst can discover obstacles to achieving system requirements. Input obstacles can be derived from scenarios to test validation routines and other functional requirements. Obstacle analysis has since been refined into a formal process for discovering the achievability of system goals with respect to a set of environmental states, taken from scenarios [28]. Scenarios, therefore, can fulfil useful roles either as test data, as a stimulant to reasoning in validating system requirements, or by providing data for formal model checking.

In HCI scenario-based methods have become an accepted approach for requirements discovery and design exploration. For example, Carroll [15] proposes scenarios of use with claims, a design rationale that represents a design principle with advantages and disadvantages. Beyer and Holtzblatt [33] argue for rich scenarios which describe a business environment as well as system use in context. Their method uses scenarios in conjunction with a suite of models analysing the business, social relationships and user tasks, whereas Carroll adopts a prototyping approach with scenarios functioning as "tools for thought".

Two RE methods have placed considerable importance on the role of scenarios, the ScenIC method [27] and SCRAM [20, 29, 34], while many other RE methods include scenarios as part of the process (e.g. [35, 24]).

ScenIC [27] proposes a schema (see figure 3) of scenario-related knowledge composed of goals, objectives, tasks, obstacles and actors. Scenarios are composed of episodes and action carried out by actors, who are usually people but may also be machines.

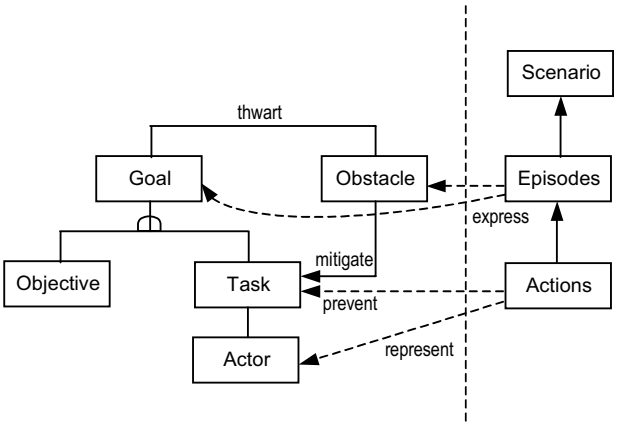


**Figure 3. Schema of scenario-related knowledge after Potts [27].**

Goals are classified into achieving, maintainting or avoiding states, while obstacles prevent goals from being achieved, or inhibits successful completion of tasks. The method proceeds in a cycle of expressing scenarios in a semi-structured format, criticising and inspecting scenarios in walkthroughs which leads to refining requirements and specifications and the next cycle. Guidelines are given for formatting scenario narratives and identifying goals, actions and obstacles. Scenario episodes are assessed with challenges to see if goals can be achieved by the system tasks, whether the actors can carry out the tasks, whether obstacles prevent the actors carrying out the tasks, etc. In this manner dependencies between goals, tasks, actors and resources can be checked to make sure the system meets its requirements.

Dependency analysis and means-ends analysis, in which tasks and the capabilities of actors are examined to ensure goals can be achieved, are also present in RE methods such as i* [17], and this illustrates the convergence of scenario- and model-based analysis in requirements engineering. Two paragraphs hardly do justice to ScenIC; however, my purpose was to draw attention to the importance of obstacle analysis, and urge the reader to consult more detail in Potts [27].

In SCRAM (Scenario-based Requirements Analysis Method), scenarios are used with early prototypes to elicit requirements in reaction to a preliminary design. The approach is based on the hypothesis that technique integration provides the best avenue for improving RE and that active engagement of users in trying out designs is the best way to get effective feedback for requirements validation. Another motivation is to use scenarios as a means of situating discussion about the design, so that new requirements can be elicited by reasoning about problems posed by scenarios describing a context of use.

This approach essentially merges the elicitation and validation role of scenarios by providing the context for the user to assess a design which itself is presenting a scenario of use. The method steps of SCRAM are illustrated in figure 4.

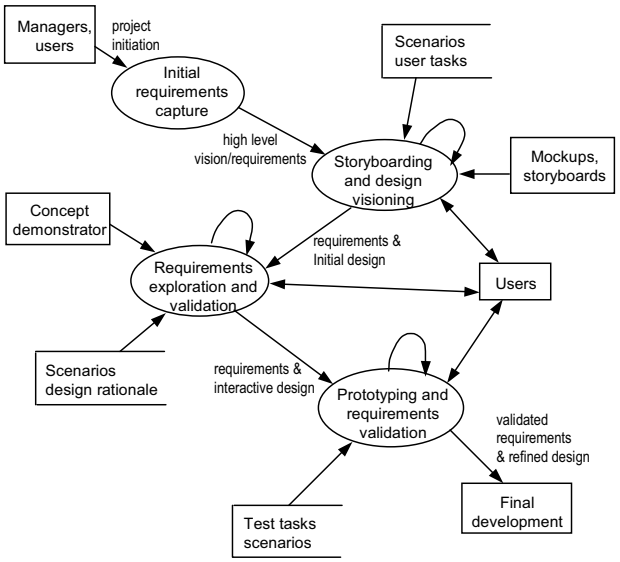


**Figure 4. Process road map of the SCRAM method.**

SCRAM does not explicitly cover modelling and specification, as this is assumed to progress in parallel, following the software engineering method of the designer's choice (e.g. UML and Unified Process [18]). The method consists of four phases:

- *Initial requirements capture and domain familiarisation*. This is conducted by conventional interviewing and fact-finding techniques to gain sufficient information to develop a first concept demonstrator. In practice this takes 1-2 client visits.
- *Storyboarding and design visioning*. This phase creates early visions of the required system that are explained to users in storyboard walkthroughs to get feedback on feasibility.

- *Requirements exploration*. This uses concept demonstrators and early prototypes to present more detailed designs to users in scenario-driven, semi-interactive demonstrations so the design can be critiqued and requirements validated.
- *Prototyping and requirements validation*. This phase develops more fully functional prototypes and continues refining requirements until a prototype is agreed to be acceptable by all the users.

The method provides process guidance for conducting walkthroughs and organising the requirements analysis process, with guidelines for interviewing and managing requirements conversations. Initial requirements capture gathers facts about the domain and captures users' high-level goals for the new system. Scenarios are elicited as examples of everyday use of the current system, with stories of problems encountered and how they are dealt with. Gathering a sufficient set of scenarios is a vexed question. There are several problems that might be encountered:

- Users tend to miss out steps in scenarios that they assume are known to the analyst: the implicit or tacit knowledge problem.
- Each person may give an individual view of problems encountered. It can be difficult to distil a set of common problems from users with diverse views.
- Acquiring a sufficient set of scenarios to cover not only normal use but also situations when things go wrong can take considerable effort. The volume of scenarios can become daunting, and this presents a problem of finding a valid sub-set.
- People tend to either forget abnormal examples or to exaggerate problems. Problems encountered most frequently and recently will be recalled first, but individuals will remember different episodes. Unravelling these potential biases can be difficult, e.g. a personality clash may make a particular problem vivid for one individual whereas for everyone else, the problem was minor.

The best way to proceed is to gather scenarios of normal system use; look for commonalties between different individual versions and create a common

"normal use case". Note that where individual variations occur, these can be useful hooks for questions later on about different individual strategies for using the system. Once the normal use case is in place, gather a set of exceptions (c.f. alternative paths in use cases). The number of alternatives necessary depends on system complexity and safety criticality. This phase also captures users' high-level goals.

Scenarios complement goal modelling because, while goals focus on abstractions that describe users' intentions, scenarios make abstract intentions clearer by giving examples of how a new system might work to fulfil users' goals. Policies and high-level aims are decomposed into lower-level goals, and scenarios of business strategy, competitors' actions and system operation can help this process. To give an example, in safety critical systems the top level policy might be "to expedite the safe navigation of the ship within the constraints of operational efficiency". The weakness of goal modelling is recording vague intentions without real thought about their practical implications. Scenarios can help by making the abstract concrete. As visions of the future system's usage, scenarios cannot be created until analysis has decomposed the system to a level where some detail of sub-goals is apparent. An example might be "the system diagnoses the potential fire hazards with different types of cargo and recommends safety measures to take if the cargo is likely to be explosive or emit noxious chemicals. The information is passed to the fire fighting crew who can take appropriate action." This scenario suggests further questions (and hence discovers further goals) about assumptions concerning the system's knowledge of the cargo – which may not be accurate –and whether the crew know what the appropriate action is (prompting a possible training requirement). Scenarios therefore have their role to play in complementing goal models that record a hierarchy of user intentions and their relationships.

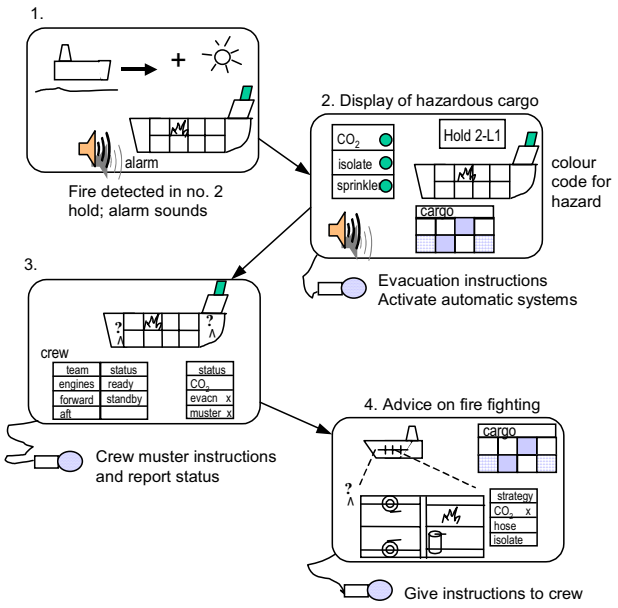


**Figure 5. Storyboard sketches for a ship emergency management system.**

Storyboards are created by developing a preliminary design from a sub-set of the usage scenarios gathered in phase 1. Storyboards are sketches or mock up screens that show key steps in user system interaction. Figure 5 illustrates a storyboard sequence derived from the scenario script. The analyst walks through the storyboard explaining what happens at each stage in terms of system functionality, and asks for the users' opinions. The limitation of storyboards is their poor interactivity. Users can be asked to simulate the actions they would carry out, but mimicking the system response is more difficult. One of the merits of storyboards and scenarios is that they help involve users in design. When users voice concerns about a design, users' reactions and suggestions for improvements are recorded. Storyboards and paper prototyping allow for quick iterations of a design, but they can mask the system functionality. Better feedback will be obtained by demonstrating an interactive prototype in the next phase of requirements exploration.

Prior to the session the concept demonstrator is developed and tested. A concept demonstrator is an early prototype with limited functionality and interactivity (see figure 6), so it can only be run as a "script". Scripts illustrate a scenario of typical user actions with effects mimicked by the designer. Concept demonstrators differ from prototypes in that only minimal functionality is implemented and the user cannot easily interact with the demonstrator. A contextual scenario is developed based on the preliminary domain analysis. This is a short narrative (half to one page) describing a situation taken from the users' work context, e.g. "a typical day in the life of ..." running through key tasks. It should also contain sufficient background material to "situate" the action, that is to give the users enough information to interpret the script. An example for a ship emergency management system follows.

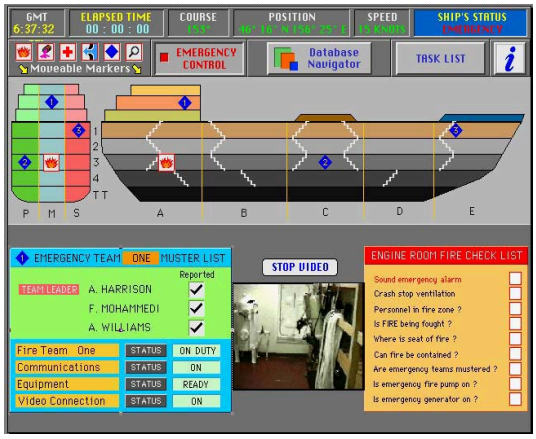

**Figure 6. Concept demonstrator for ship emergency management system.**

*Contextual scenario*

Your ship is a modern container ship of 30,000 tons displacement with a multinational (mainly Filipino) crew of 36 with five UK officers. The cargo manifest lists containers with a mixture of industrial goods and domestic removals. You have left Southampton at 10 a.m. this morning en route to Cape Town and are proceeding west down the English Channel heading 250 degrees at 15 knots, position 50 miles south of Plymouth. The weather is fine, visibility range 15 miles, wind NW force 3. At 3.10 a member of the crew reports smoke in number two hold, and a fire alarm sounds.

A *scenario script*, based on the captain's emergency management task, is used for the concept demonstrator.

1. The location of the fire is investigated and preliminary instructions given to the crew to evacuate the area if necessary.
2. Automatic fire suppression systems are activated if present, such as flooding compartments with $CO_2$.
3. Instructions are given by tannoy to the crew to proceed to fire muster stations and start to fight the fire.
4. Junior officers are assigned to manage key fire fighting teams.
5. The cargo manifest is checked to see if any dangerous (explosive, flammable, corrosive, etc) cargo is present near the fire.
6. Fire fighting tactics are planned to account for any dangerous cargo and other hazards, e.g. electrical equipment.
7. Instructions are given to fire fighting crews. The progress of fire fighting is monitored and further instructions given as necessary, until the fire is under control.

Note that the scenarios do not attempt to cover all aspects of the users' tasks or the situation; for instance the damage assessment phase is not described, nor is the means of communication between the captain and crew. A number of validation sessions may be necessary to test different parts of the design.

Probe questions are asked at key points in the demonstration script. The users are invited to critique the concept demonstrator. The concept demonstrator is explained by the analyst, while another member of the design team runs and interacts with the system. Limited hands-on testing may be provided at the end of the session. In a follow-up phase, the users are encouraged to clarify any points they found ambiguous, go back to any parts of the demonstration, and elaborate further requirements. The requirements engineers may also follow up points raised or user comments made during the session.

Scenarios can be linked to design decisions represented in design rationale diagrams that illustrate trade-offs and assumptions affecting choice (see figure 7). Further links can complete the pathway from scenarios to more formal requirements specifications.

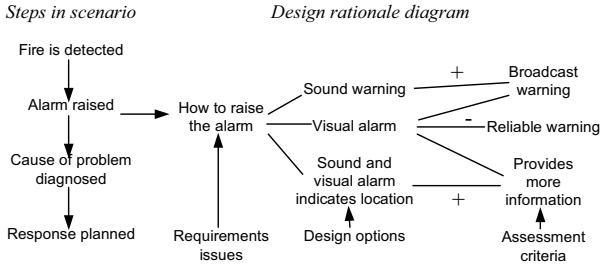


**Figure 7. Design rationale diagram illustrating alternative design solutions for requirements issues linked to the concept demonstrator scenario script.**

Once the demonstration has been completed, a summary of the requirements is listed on a whiteboard (or another appropriate medium). The requirements are discussed and prioritised using an essential/ useful/optional scale. Design rationale may be introduced in this step to help structure discussion about design trade-offs with assessment criteria (often non-functional requirements) that can be used to judge the merits of alternative solutions. The process can be repeated using a more functional prototype as necessary.

## 6. Tool support for scenario-based RE

While commercial requirements management tools such as DOORS and Rationale Rose enable scenarios to be recorded either explicitly or in general narrative comment fields, there are few tools which support the process of scenario-based RE. Formatting and checking scenarios for consistency can be helped by lexicial approaches which provide a database of keywords and templates for formatting scenario-related knowledge [36]. The RETH hypertext tool [10] links scenarios, goals and functional requirements to support scenario inspections. The CREWS-SAVRE tool [37] helped the generation of variations on a seed scenario by first expanding possible event sequences that could be traced from a use case-like behaviour model, then suggesting possible permutations

to an event sequence using a taxonomy of errors drawn from the human factors literature [38, 39].

Scenario-based requirements validation has been explored using high-level descriptions of systems expressed as Bayesian Belief Nets (BBNs) [40]. Scenarios expressed as task-based episodes in a similar manner to ScenIC are evaluated with a BBN tool that automatically tests variations in the scenario for environmental conditions such as weather, and training of human actors. This approach was taken further by applying evolutionary computing techniques to generate variations in a requirements specification, then running the specifications against a set of scenarios, determining which design variation had better performance and breeding these variations following the principles of evolution [41]. In spite of these initiatives, support for scenario-based RE is still primitive. Tools are needed that automatically extract interesting facts from scenarios, to produce models which can then be checked for consistency, completeness etc.

## 7. Reflections and outstanding problems

While scenarios are an important and useful addition to the battery of RE techniques they are not without problems. The two most critical problems are sampling and coverage. Both reflect the tension between specific detail in scenarios and abstraction in requirements specifications. Sampling is difficult because of representativeness, i.e. how do you know when you have collected an appropriate and representative set of scenarios for the current problem? This is linked to coverage: how do you know when you have a sufficient set of scenarios for adequate testing in requirements validation? There are no easy answers to these problems. One approach is to generate variations from a single seed scenario [29, 34, 42] by using a schema to suggest variation points. However, this assumes a formatted scenario which is closer to a model; furthermore, automatic generation creates too many scenario variants which swamp the requirements engineer in excessive detail [16]. The silver bullet of scenario-based RE is the 20/20 foresight, or how to anticipate critical aspects in a future system environment that will impact on system requirements. Of course that doesn't exist otherwise governments (e.g. socio-economic scenarios), the military (e.g. wargame scenarios) and manufacturers of complex systems (e.g. scenarios for avionics systems) would not experience the unexpected. Creative brainstorming and reuse of knowledge can improve the practice of scenario-based RE. Further research needs to be undertaken to investigate the dependencies between scenarios and models at different levels of granularity from enterprise business models and competition scenarios [43] to requirements for designed systems that we are familiar with in RE.

There are approaches to eliciting and generating better sets of test scenarios besides brute force validation by volume. One approach is to use constraint relaxation having started with worst case scenarios, and this can be partially automated by converting scenarios into event sequence models and then running these against a requirements specification [40].

While scenarios have become an established technique in RE and elsewhere, many questions remain for future research. The trade-off between informal scenarios as tools for thought and more formal scenarios which converge with models may vary between domains. In some cases scenarios are throw-away examples to stimulate thought; in other cases scenarios become transformed into models by a systematic process. The process of extracting knowledge from and testing with scenarios is still in its infancy. Furthermore, scenarios lie on the boundary of informal and formal representations of knowledge. The bottleneck is human ability to process large volumes of narratives and examples. In the future, tools that extract information from speech, text and image may play an increasingly important role in scenario-based RE.

## References


[1] M. Jarke, X.T. Bui and J.M. Carroll, "Scenario Management: An Interdisciplinary Approach," *Requirements Engineering*, vol. 3, 155-173, 1998.

[2] C. Rolland, C.B. Achour, C. Cauvet, J. Ralyte, A.G. Sutcliffe, N.A.M. Maiden, et al., "A Proposal for a Scenario Classification Framework," *Requirements Engineering*, vol. 3, 23-47, 1998.

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

[4] K. Kuutti, "Workprocess: Scenarios As a Preliminary Vocabulary," in *Scenario Based Design*, J.M. Carroll, Ed. New York: Wiley, 1995, 19-36.

[5] A.I. Anton and C. Potts, "A Representational Framework for Scenarios of System Use," *Requirements Engineering*, vol. 3, 219-241, 1998.

[6] A.I. Anton and C. Potts, "The Use of Goals to Surface Requirements for Evolving Systems," *1998 International Conference on Software Engineering: Forging New Links*, 1998, 157-166, Los Alamitos CA: IEEE Computer Society Press.

[7] M. Kyng, "Creating Contexts for Design," in *Scenario Based Design*, J.M. Carroll, Ed. New York: Wiley, 1995,

85-108.

[8] C. Potts, K. Takahashi and A.I. Anton, "Inquiry-Based Requirements Analysis," *IEEE Software*, vol. 11, 21-32, 1994.

[9] P. Heymans and E. Dubois, "Scenario-Based Techniques for Supporting the Elaboration and Validation of Formal Requirements," *Requirements Engineering*, vol. 3, 1998.

[10] H. Kaindl, "An Integration of Scenarios with Their Purposes in Task Modelling," *DIS 95 Conference Proceedings*, 1995, 227-235, New York: ACM Press.

[11] A. Cockburn, *Writing Effective Use Cases*, Boston MA: Addison-Wesley, 2001.

[12] I. Alexander, "Initial industrial experience of misuse cases in trade-off analysis", *Proceedings IEEE Joint International Conference on Requirements Engineering*, 2002, 61-70, Los Alamitos CA: IEEE Computer Society Press.

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

[14] P.A. Gough, F.T. Fodemski, S.A. Higgins and S.J. Ray, "Scenarios: An Industrial Case Study and Hypermedia Enhancements," *1995 IEEE International Symposium on Requirements Engineering (RE '95)*, 1995, 10-17, Los Alamitos CA: IEEE Computer Society Press.

[15] J.M. Carroll, *Making Use: Scenario-Based Design of Human-Computer Interactions*, Cambridge MA: MIT Press, 2000.

[16] A.G. Sutcliffe, N.A.M. Maiden, S. Minocha and D. Manuel, "Supporting Scenario-Based Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 24, 1072-1088, 1998.

[17] J. Mylopoulos, L. Chung and E. Yu, "From Object-Oriented to Goal-Oriented Requirements Analysis," *Communications of the ACM*, vol. 42, 31-37, 1999.

[18] Rational Corporation, *UML: Unified Modelling Language Method*, [http://www.rational.com], 1999.

[19] J. Conklin and M.L. Begeman, "GIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Transactions on Office Information Systems*, vol. 64, 303-331, 1988.

[20] A.G. Sutcliffe, "Scenario-Based Requirements Analysis," *Requirements Engineering*, vol. 3, 48-65, 1998.

[21] A.G. Monk and P. Wright, *Improving Your Human-Computer Interface: A Practical Technique*: Prentice Hall, 1993.

[22] A.G. Sutcliffe, "Bridging the Communications Gap: Developing a Lingua Franca for Software Developers and Users," *INFORSID*, 2000, 13-32, Toulouse: Inforsid.

[23] A.G. Sutcliffe and J.M. Carroll, "Generalizing Claims and Reuse of HCI Knowledge," *BCS-HCI Conference*, 1998, 159-176, Berlin: Springer-Verlag.

[24] I. Sommerville and G. Kotonya, *Requirements Engineering: Processes and Techniques*, Chichester: Wiley, 1998.

[25] P. Dubois, E. Dubois and J. Zeippen, "On the Use of a Formal Representation," *ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, 1997, 128-137, Los Alamitos CA: IEEE Computer Society Press.

[26] C.L. Heitmeyer, R.D. Jeffords and B.G. Labaw, "Automated Consistency Checking of Requirements Specifications," *ACM Transactions on Software Engineering and Methodology*, vol. 5, 231-261, 1996.

[27] C. Potts, "ScenIC: A Strategy for Inquiry-Driven Requirements Determination," *4th IEEE International Symposium on Requirements Engineering*, 1999, 58-65, Los Alamitos CA: IEEE Computer Society Press.

[28] A. Van Lamsweerde and E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 26, 978-1005, 2000.

[29] A.G. Sutcliffe, *User-Centred Requirements Engineering*, London: Springer-Verlag, 2002.

[30] P.N. Johnson-Laird, *The Computer and the Mind: An Introduction to Cognitive Science*, Cambridge MA: Harvard University Press, 1988.

[31] J.M. Carroll, Ed. *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, San Francisco: Morgan Kaufmann, 2003.

[32] J.P. Djajadiningrat, W.W. Gaver and J.W. Frens, "Interaction Relabelling and Extreme Characters:

Methods for Exploring Aesthetic Interactions," *DIS2000 Designing Interactive Systems: Processes, Practices Methods and Techniques*, 2000, 66-71, New York: ACM Press.

[33] H. Beyer and K. Holtzblatt, *Contextual Design: Defining Customer-Centered Systems*, San Francisco: Morgan Kaufmann, 1998.

[34] A.G. Sutcliffe and M. Ryan, "Experience with SCRAM: A SCenario Requirements Analysis Method," *IEEE International Symposium on Requirements Engineering: RE '98*, 1998, 164-171, Los Alamitos, CA: IEEE Computer Society Press.

[35] J. Robertson and S. Robertson, *Mastering the Requirements Process*, Harlow: Addison Wesley, 1999.

[36] J. Leite, G.D.S. Hadad, J. HoracioDoorn and G.N. Kaplan, "A Scenario Construction Process," *Requirements Engineering*, vol. 5, 38-61, 2000.

[37] N.A.M. Maiden, S. Minocha, K. Manning and M. Ryan, "CREWS-SAVRE: Systematic Scenario Generation and Use," *IEEE International Symposium on Requirements Engineering: RE '98*, 1998, 148-155, Los Alamitos CA: IEEE Computer Society Press.

[38] E. Hollnagel, *Human Reliability Analysis: Context and Control*, London: Academic Press, 1993.

[39] J. Reason, *Human Error*, Cambridge: Cambridge University Press, 1990.

[40] A.G. Sutcliffe and A. Gregoriades, "Validating Functional System Requirements with Scenarios," *IEEE Joint International Conference on Requirements Engineering*, 2002, 181-188, Los Alamitos CA: IEEE Computer Society Press.

[41] A.G. Sutcliffe, "Evolutionary Requirements Analysis," *IEEE Joint International Conference on Requirements Engineering*, 2003,Los Alamitos CA: IEEE Computer Society Press.

[42] A.G. Sutcliffe, J.E. Shin and A. Gregoriades, "Tool Support for Scenario-Based Functional Allocation," *21st European Conference on Human Decision Making and Control*, 2002,.

[43] X.T. Bui, G. Kersten and P.C. Ma, "Supporting Negotiation with Scenario Management," *29th Hawaii International Conference on System Sciences*, 1996, 209-219, Honolulu: University of Hawaii.