Available online at www.sciencedirect.com

ScienceDirect

Information and Software Technology 50 (2008) 198–220

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Scenario support for effective requirements

Thomas A. Alspaugh [a,*], Annie I. Antón [b]

[a] Department of Informatics, Donald Bren School of Information and Computer Sciences, University of California, Irvine, CA, USA
[b] Department of Computer Science, North Carolina State University, College of Engineering, 1010 Main Campus Drive (EGRC 408), Raleigh, NC 27695-8206, USA

Received 19 December 2005; received in revised form 14 December 2006; accepted 30 December 2006
Available online 10 January 2007

Abstract

Scenarios are widely used as requirements, and the quality of requirements is an important factor in the efficiency and success of a development project. The informal nature of scenarios requires that analysts do much manual work with them, and much tedious and detailed effort is needed to make a collection of scenarios well-defined, relatively complete, minimal, and coherent. We discuss six aspects of scenarios having inherent structure on which automated support may be based, and the results of using such support. This automated support frees analysts to concentrate on tasks requiring human intelligence, resulting in higher-quality scenarios for better system requirements. Two studies validating the work are presented.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Requirements engineering; Scenario analysis; Scenario management

1. Introduction

Good requirements are critical: stakeholders with good requirements are more likely to get what they want, and developers with good requirements are more likely to finish in a reasonable time, or finish at all. Scenarios can be good requirements in this sense. In our experience, scenarios are helpful to stakeholders, who can understand them and relate them to their own experiences; they are helpful to requirements analysts, because everyone can discuss, negotiate, and plan using a single kind of artifact; and to developers, for whom scenarios are concrete and specific examples of system behaviors [16,17,20]. However, to answer any interesting question about prose scenarios requires careful reading, hard thought, insight, and skill. We have found that analysts are faced with many tedious, time-consuming tasks in making high-quality scenarios, and need automated support for these tasks or they would not get done – the extra time it takes to produce high-quality scenarios is often just not available. Scenarios are fundamentally informal, but have some structure and consistent form and meaning that can be the basis for automated support. Tool support for tasks that *can* be automated will free analysts for other tasks that require intelligence and judgement so they can make better scenarios that will support successful development of software that satisfies its stakeholders [5,9].

In this paper, we describe an approach for supporting more effective work with scenarios. The guiding principle is that although scenarios are fundamentally informal and analysts must perform many tasks manually, there are aspects of scenarios that have an implicit structure and meaning and these can be the basis for automated support. Using tools to do tedious, exacting, or uninteresting tasks helps eliminate human errors by releasing people to concentrate on more interesting work for which human intelligence is essential and most valuable. The result is scenarios that more effectively explore the space of possibilities, more accurately convey the stakeholders' and analysts' understanding, needs, and tradeoffs, and more effectively highlight the requirements issues that can and

[*] Corresponding author. Tel.: +1 949 824 7355.
E-mail address: alspaugh@ics.uci.edu (T.A. Alspaugh).

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2006.12.003

should be resolved in terms of requirements rather than being left for developers to sort out in subsequent phases.

1.1. Scenarios as requirements

Scenarios and use cases are widely used in a number of ways during the development process, and by a variety of participants [2,21]. Stakeholders (customers, users, and others affected by a system) use them to communicate what is wanted, and developers (designers, programmers, testers, etc.) use them to confirm their understanding [21,47,55]. They may be the primary form in which requirements are recorded [31,37,39], a preliminary form from which specialists produce more refined forms such as goals and requirements [43], or a guide and scaffolding in a process by which other artifacts such as goals are developed [56]. They are used to simulate and explore a system's use [21] or a design's utility [60], to present a test or validation [19], and to derive tests from requirements [8].

Scenarios are traditionally not considered to be a form of requirements, and many researchers and practitioners still hold this view. However, in our experience it is becoming more and more common for scenarios (or use cases composed of scenarios) to be a significant part of a system's requirements, or indeed to be the system's requirements. The Euronet software requirements document, discussed below, is an example [18]. This document, entitled "Software requirements specification", consists of a set of use cases plus an incomplete group of screen layouts; here the use cases *are* the requirements and are identified as such. More and more industrial practitioners are developing and working from similar requirements.

In all these uses, the quality of the scenarios (or the use cases containing them) is a key factor in the success or failure of the system development, especially when the scenarios are used as the system requirements. We need to help practitioners create scenarios that are easier to manage and analyze, and that can act as effective requirements that support efficient and successful software development as well as other uses of requirements. Although high-quality requirements do not guarantee efficient development of an acceptable product, low-quality requirements increase the likelihood that development will be inefficient, the product will not be acceptable, or that the development process will not produce a system at all. One survey of some 8000 US information technology development projects in 1994 found that a "clear statement of requirements" was cited as the most important factor for 13% of successful projects. Incomplete or changing requirements were cited as the most important factors for 24% of projects that produced unsatisfactory systems or were late or over budget, and for 22% of projects canceled before a system was produced [59]. The importance of requirements quality as a factor in the success or failure of system development has been a constant for as long as software development has been studied [24–26,30,33–35].

1.2. Four scenario qualities

In this work, we aim for four specific qualities that affect the efficiency and success of scenarios in system development.

Well-definedness:	*Well-defined scenarios use terms that are clearly understood by their readers, define terms and references that may be unfamiliar or are unique to the system in question, and resolve conflicts and contradictions that must be addressed while remaining appropriately abstracted from inessential choices and details [*9,13,27,32,65*].*
Coverage:	*A scenario collection is complete enough if it covers stakeholders' needs and explores the issues and tradeoffs that if left unresolved would cause more delay or rework later [*16,53*].*
Minimality:	*A minimal scenario collection gives one description for each behavior and one definition for each individual concept or term. Duplicates are eliminated and unnecessary redundancy is reduced [*16*].*
Coherence:	*The scenarios of a coherent collection support and reinforce each other. They use the same terms to indicate intentional reuse of concepts, and the same episodes to indicate intentional reuse of events; the contexts and results the scenarios create in the world are used by other scenarios; and they suggest ways of composing scenarios in sequence to achieve larger results than individual scenarios provide. We have found this quality to be very important from our experiences in working with scenarios as an industrial developer or researchers working with developers, over a period of over 20 years.*

In our experience, these qualities support the resolution of questions, tradeoffs, and conflicts in terms of requirements whenever that is possible, and the production of a requirements specification that increases the chances of successful development. Lack of any or all of these factors causes extra and more expensive work in later phases that potentially could have been avoided by resolving the issue in terms of the requirements scenarios during the requirements phase.

1.3. Six avenues for automated support

In this paper, we present six techniques based on the underlying structure and meaning of scenarios:

- using glossaries to define attributes and their values (Section 3.1);
- using episodes to express dependency between scenarios (Section 3.2);
- constructing the episode reference diagram as a means of understanding a scenario collection and highlighting possible problems (Section 3.3);
- identifying shared events to appropriately define episodes (Section 3.4);
- specifying the context of each scenario directly in terms of the contexts of other scenarios (Section 3.5). and
- calculating similarity among scenarios as a basis for establishing minimality, and a basis for searching for specific scenarios (Section 3.6).

These techniques mutually reinforce each other, are amenable to automated support, and form the components of our approach for more effective work with scenarios.

1.4. Roadmap

The remainder of the paper is organized as follows. Section 2 defines common terminology and background concepts. Section 3 presents the six techniques of our approach for improving scenario quality and providing effective automated support for work with scenarios. Section 4 discusses our tool, SMaRT, that automates some components of the approach. Section 5 describes two studies showing the effectiveness of our approach: EMS (Enhanced Messaging System), a voice mail system used by BellSouth Telecommunications to prototype new features for its products; and Euronet, a quote management system developed and used internally by Asea Brown Boveri (ABB). We give examples from these throughout the paper. Section 6 summarizes related work, and Section 7 discusses our results and lists future work.

This paper amalgamates and extends work reported in conference and workshop papers [4–6].

2. Terminology and background

2.1. Terminology

We define the following key terms.

- A *scenario* is a sequence of events, plus possibly some associated attributes such as goals, requirements, viewpoint, author, and pre- and postconditions [6]. Common scenario representations can be abstracted to this form [14]. See Fig. 1 for an example scenario.
- An *event* consists of an actor-action pair. An actor may be a specific person, component, or system, or may be an unbound role or parameter that can be filled by any of several specific actors. The action takes place over some time interval.
- An *episode* is a scenario that is used as an event of several scenarios.
- A *context* is a state of the world needed by or created by a scenario or an event. It may include the presence of specific resources and participants, their physical location, and results of other occurrences that are temporally or causally related to it.

2.2. Scenario structure

Structure refers to the way the constituent parts are put together to make a whole, and the mutual relations of those parts in determining the character of the whole. Scenarios contain distinct types of information (events, goals, conditions, etc.) that can be expressed as attributes and attribute values. A scenario describes its events as they occur in a total or partial temporal sequence. Many events, if examined closely, are composed of subevents, sometimes in a sequence, sometimes iterated, sometimes as a choice among alterna-

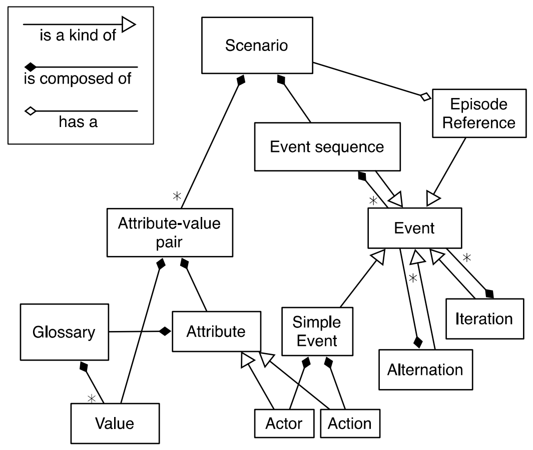


Fig. 2. Scenario structure meta-model.

S38. Caller calls EMS directly and leaves a message

Requirements:	R4.1.
Precondition:	Has (n) New, Connected.
Postcondition:	Has (n+1) New, Left Message, Connected.

1. A caller calls EMS directly and chooses the "Leave message" menu item.
2. EMS asks the caller to enter a subscriber's telephone number.
3. The caller dials the subscriber's telephone number.
4. EMS plays the subscriber's name and announcement.
5. The caller leaves a message.

Fig. 1. EMS scenario S38 "Caller calls EMS directly and leaves a message".

tives. We use this scenario structure and meaning, outlined in Fig. 2, as the basis of our approach.

2.3. Inter-scenario relationships

While individual scenarios are effectively represented as a sequence or narrative, a collection of scenarios does not naturally have a sequential form. The relationships among scenarios are often partial orders or more complexly structured, and are most effectively represented as graphs. Some examples are:

- the *episode relation* that relates a scenario specifically to the episodes it uses (Fig. 6);
- the *shared event relation* that relates a scenario to other scenarios sharing one or more of the same events (Fig. 7); and
- the *context relation* that relates a scenario to those it can precede, follow, occur with, or substitute for because of the context it uses or produces (Fig. 11).

Some of these relationships are implicit in the scenarios. The relationship collates and presents this dispersed information coherently in one place. The episode relation is an example. All the information it expresses is available in the scenarios (and in fact the relation can be derived automatically from scenarios whose structure is formalized as in SMaRT), but the information is much easier to see and reason about in a single diagram.

Other relations are absent or only partially specified in the scenarios. To express and work with these relationships, we must augment the scenarios with additional information. The context relation is an example. If one scenario's postcondition implies a second scenario's precondition, then the first scenario can be said to establish the context for the second. If two scenarios' preconditions are logically equivalent, the two scenarios can be said to potentially occur together because the same context allows either. However, there are other possibilities the conditions do not express: the necessary context may be established inside a scenario rather than at its conclusion, or two scenarios may be intended as mutually exclusive alternatives, rather than potentially concurrent. Also, in our experience the conditions are often only approximations, not written to imply the required context relations between scenarios, and the additional work to make them do so and keep them that way while the scenarios evolve could be better spent expressing the desired relationships directly. We augment the scenarios with additional information in the form of context diagrams, as we discuss in Section 3.5.

3. Components of approach

3.1. Glossaries

The benefits associated with consistency checking in requirements specifications are noted by Heitmeyer et al. [38]. In four of the studies discussed by Weidenhaupt et al., project glossaries helped stakeholders establish a common understanding of scenario terms [63]. In an industrial study in which 88 scenarios were used to define requirements, stakeholders also noted the need to define general and domain-specific terms [36].

Our glossaries specify the set of values each attribute can have and define each value. Each attribute is given its own glossary of values. This simplifies the work of determining if two attribute values are the same: if the values are given by the same glossary entry, they are the same, and otherwise they are distinct. At the same time, the glossaries provide a connection back to the scenario's original prose form. We also support glossaries of specialized terms used

"Requirement" glossary:

R4.1	*(EMS shall play the subscriber's name and announcement to each caller.)*

"Condition" glossary:

Connected	*(The caller is connected to the EMS.)*
Has (m) New	*(The subscriber has m new messages and $m \geq 0$.)*
Left Message	*(The caller has left a message but has not yet hung up.)*

"Actor" glossary:

Caller	*(A person (or automatic device) who has called a subscriber's voice mailbox.)*
EMS	*(The Enhanced Messaging System.)*

"Action" glossary:
Ask the caller to enter a subscriber's telephone number
Call EMS directly and choose the "Leave message" menu item
Dial the subscriber's telephone number
Leave a message
Play the subscriber's name and announcement

"Event" glossary:
(Caller, Call EMS directly and choose the "Leave message" menu item)
(Caller, Dial the subscriber's telephone number)
(Caller, Leave a message)
(EMS, Ask the caller to enter a subscriber's telephone number)
(EMS, Play the subscriber's name and announcement)

Glossary of other defined terms:

announcement	A subscriber's announcement is a recording that a caller hears when he or she reaches the subscriber's voice mailbox.
subscriber	The subscriber is a person for whom EMS provides a voice mailbox.

Fig. 3. Extracts from EMS attribute and term glossaries.

in the text. Fig. 3 shows attribute-value and term glossaries for the EMS scenario "Caller calls EMS directly and leaves a message" (Fig. 1).

The use of glossaries exemplifies the quality of well-definedness, and also has positive effects that are less direct. A glossary encourages scenario authors to move the work of clearly defining each term out of the main text and into the glossary. Then each term need only be made clear once, rather than at each use. A reader can read confidently, knowing he or she can look up any unfamiliar term. Glossary support encourages reuse of already-defined terms as appropriate, instead of one or more new terms with the same or nearly the same definition. A smaller number of more-distinct concepts are used, rather than a larger set of concepts not clearly distinguished from each other. All this results in efficiencies for authors and readers, and helps achieve a minimal and coherent scenario collection.

3.2. Episodes

Episodes, sequences of events intentionally shared by two or more scenarios, play a significant role in scenario management [53]. To manage episodes, a system must be able to distinguish events and recognize those that are identical. Two events are identical if they share the same actors and action. The set of actors for a system is well-defined; hence, a tool can readily recognize identical actors in two or more events. However, the set of actions may not be so well-defined. If so, the intervention of an analyst is needed to recognize identical actions.

We envision several ways to support the identification of identical actions. In each case presented below, the tool presents pairs of potentially identical actions, the analyst determines whether they are identical or distinct, and the tool records the decision for future use.

1. *The tool presents only the pairs of actions that the analyst asks for.* The analyst chooses which pairs to examine, but for a thorough comparison, many pairs must be examined.
2. *The tool presents potentially identical pairs of actions whose textual descriptions are similar*. The analyst may have to examine many pairs, but fewer than in the first method.
3. *The tool presents only potentially identical pairs of actions that, if identical, would make two events identical.* The textual descriptions of the actions must be similar, and the actions must appear in two events sharing the same actor. The analyst examines fewer pairs and many actions will not need to be examined. The analyst only need examine the pairs that might result
4. *The tool presents only potentially identical pairs of actions that, if identical, might make two subsequences of n or more events identical*. In addition to the three previous conditions, the events that the actions appear in must be part of two shared subsequences of length $n \geqslant 2$ that are potentially identical. n may be adjusted to reduce the number of presented pairs as desired.

Once episodes are identified and recorded in the scenario database, several helpful features are possible:

1. Display of an episode in a scenario either as the episode name only, or expanded with the episode's events visible inline in the scenario.
2. Visible marking of episodes when the events of a scenario are displayed.
3. Links from each scenario to others sharing the same episode.
4. Display of the scenarios that share an episode either individually, or simultaneously with one shown as a variant of the other(s).
5. A warning when an event in an episode is edited.
6. Presentation of a choice when editing events in an episode: "change all the scenarios that share this episode", or "detach this scenario from the others and change it only, leaving the others the same".

The value of these features increases with the size of a project. As the numbers of both scenarios and developers increase, it becomes more difficult to track scenario redundancy and dependencies. Episodes explicitly represent these relationships, allowing analysts to monitor and control inter- and intra-scenario evolution. In addition, managing decisions about scenario consistency and dependency can prevent and reduce errors in the scenario database. This early prevention of errors can save money and time in the requirements analysis process: the earlier errors are caught, the easier and more inexpensive they are to fix. Thus, our episode management strategy improves requirements analysis by

Event in				
S2	S3	S4	Actor	Action
1.	—	—	Subscriber	Dial EMS from some his or her telephone
—	1.	—	Subscriber	Dial EMS from some unsubscribed telephone
—	—	1.	Subscriber	Dial EMS from another subscriber's telephone
2.	—	—	EMS	Answer and announce the subscriber's name
—	2.	—	EMS	Answer with some announcement
—	—	2.	EMS	Answer and announce the other subscriber's name
—	3.	3.	Subscriber	Dial his or her telephone number
—	4.	4.	EMS	Announce the subscriber's name
3.	5.	5.	Subscriber	Dial his or her passcode
4.	6.	6.	EMS	Authenticate the passcode

Fig. 4. Possible EMS episodes in S2, S3, and S4.

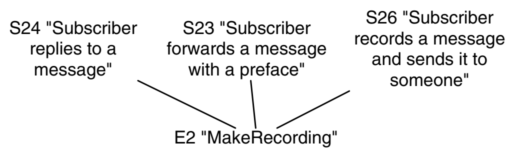

S24 "Subscriber replies to a message"
S23 "Subscriber forwards a message with a preface"
S26 "Subscriber records a message and sends it to someone"
E2 "MakeRecording"

Fig. 5. Episode reference diagram for EMS.

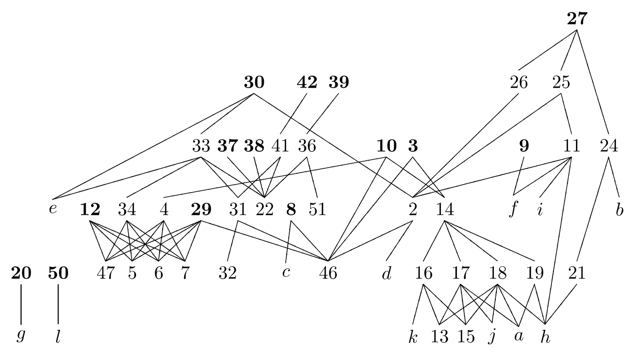


Fig. 6. Episode reference diagram for the ABB Euronet scenarios.

partially automating event comparisons, error prevention, and dependency tracking, resulting in a faster, more reliable scenario development process.

Table 4 shows potential episodes shared by EMS scenarios S3 and S4, and by S2, S3, and S4. Each scenario's events are listed with their event numbers in that scenario. We see that events 3–6 are shared by S3 and S4, and could be abstracted as an episode shared by both scenarios, as could S2's events 3 and 4 and S3 and S4's events 5 and 6.

3.3. Episode reference diagrams

An episode reference diagram shows which scenarios use other scenarios as episodes. Each node in the diagram represents a scenario, and each edge connects a scenario to another scenario, lower in the diagram, that is an episode of the first one. Fig. 5 shows the episode references for the 32 EMS scenarios. These scenarios have a simple and straightforward reference pattern, and most of the scenarios do not use an episode. Fig. 6 shows the episode reference diagram for the 64 Euronet scenarios, which use each other as episodes to a startling degree.

An episode reference diagram is useful in several ways:

3.3.1. A compact and informative view

Episode reference diagrams show how scenarios share episodes, in a compact and informative way. It takes some work to extract this information from the scenarios, because the connections are dispersed among the scenarios and the larger relationships are not obvious from simply reading the scenarios. An episode reference diagram presents these relationships all at once, clearly and in a small space.

3.3.2. Identifying dependencies

Each episode shared by two or more scenarios represents a dependency between those scenarios [6]. In the center of Fig. 6 we see that episode 46 is shared among the six scenarios 2, 3, 8, 10, 29, and 31. These scenarios are therefore related, and a change to one is likely to necessitate changes in some or all the others. Any change to episode 46 changes the others, as its events are part of each of them. A change to any of them may result in changes to the others, to the extent that they are coupled by the shared events of the episode and the shared rationale for its use. If the overall design evolves and reduces the coupling among them, the broad sharing of episode 46 may end, either with a new episode similar to 46 that replaces it as an episode in some of the scenarios, or by independent in-line event sequences that provide a customized replacement for 46 in some or all of the scenarios. The evolution of the dependencies changes the episode references, and is reflected visually in the diagram as it evolves to match.

3.3.3. Identifying multiple dependencies

An episode reference diagram shows the relationships for more than one episode at a time, and is particularly useful for showing how scenarios are simultaneously related through more than one episode. An example of this is scenarios 16 and 17 in the lower right of Fig. 6, which are related through their episodes 13 and 15, and 17 and 18 which are related through all three of their episodes 13, 15, and *j*. The diagram enables analysts to consider dependencies resulting from several episodes at once, and to quickly trace chains of references through several episodes.

3.4. Shared events

The same events can occur in several scenarios, whether because the task they describe is needed for the work of those scenarios, because the context they need or produce occurs at those points in the scenarios, or for other reasons. Events shared among scenarios often indicate a dependency between the scenarios, so that a change in one necessitates a change in the others. Analysts can create episodes from the shared events to make the sharing explicit, but only if they are aware the sharing exists. The sharing is implicit in the scenario text but is not obvious in that form, especially for a large collection of scenarios.

A *shared event diagram* shows which events recur in which scenarios. Fig. 7 shows a shared event diagram for the EMS scenarios. Each shared event is shown as two or more numbered rectangles (such as 3 in the top row and 5 in the row below) connected by a heavy doubled line. Each of the rectangles represents an instance of the event in a particular scenario (such as event 3 in scenario S2, in the top row). It is joined to the other instances of the same event (as S2's event 3 is joined to S3's event 5, and on to S4's event 5). See Fig. 4 for the text of S2, S3, and S4. Sharing of an episode involves sharing all the episode's events as a single event in two or more other scenarios; this is indicated on the right side of the diagram by the events (S23's 2.2.3.1, S24's 2, and S26's 3) joined to the circle at the bottom that represents episode E2.

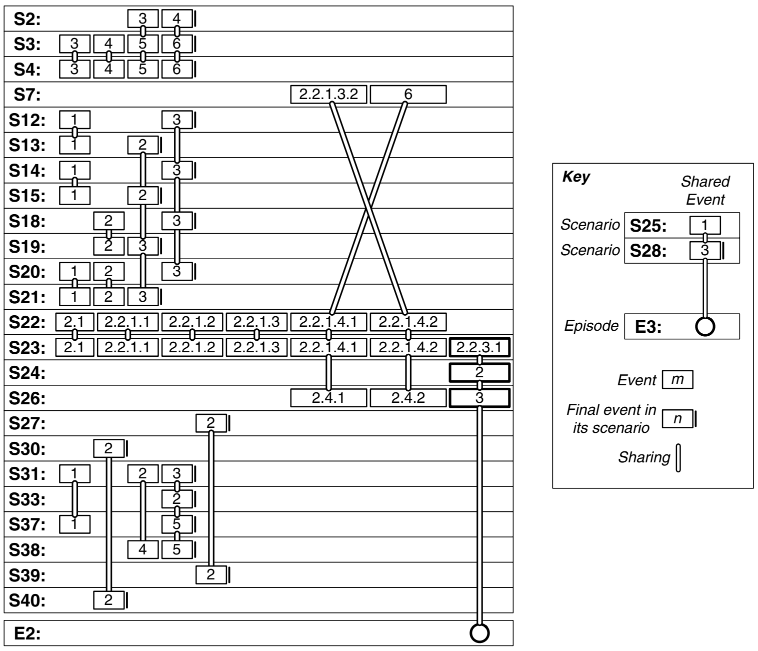


Fig. 7. Shared events diagram for EMS scenarios.

The example diagram links events that are textually identical in two or more scenarios. There is also the case of events that are similar and may have the same meaning, but are not worded identically. In order for the diagram to be the most informative, analysts first need to identify such similar events and then make them textually identical if they describe the same event. Some simple techniques for identifying similar events are sorting events alphabetically and examining the text (as was done in the EMS study), or dividing events into actors and actions, and sorting those individually (as was done in the Euronet study and supported by SMaRT). More sophisticated techniques are also possible with appropriate tool support.

A shared event diagram is useful in several ways:

3.4.1. Episode creation

Earlier we discussed tool support for identifying equivalent events. The second step is to decide whether the events shared among scenarios justify creation of an episode containing them. The shared event diagram shows the extent to which specific events are shared, and simultaneously points to the contexts in which they are shared. This extra information helps analysts make better and quicker decisions on creating episodes.

3.4.2. Working with suffix–prefix overlaps

If the last few events of one scenario are shared as the first few events of another (see Fig. 8), the analyst may have intended that the second scenario is intended to follow the first, starting where the overlap begins. Even if the scenarios were not created with that intent, if the events of the first scenario, up to the overlap, create the context needed by the second scenario, it is possible that the two scenarios occur in tandem. Such an overlap indicates the scenarios may benefit from further analysis.

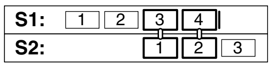


Fig. 8. One scenario's suffix overlapping with another's prefix.

3.4.3. Working with shared suffixes or shared prefixes

If two scenarios share a suffix, it may mean the events they describe create equivalent contexts in the world (see Fig. 9). This is the case with EMS scenarios S2 "Subscriber authentication" and S3 "Subscriber authentication from unsubscribed telephone" whose overlap is illustrated. Similarly, when two scenarios share a prefix, it may mean their events require equivalent contexts, as is the case with EMS scenarios S20 "Subscriber archives a message" and S21 "Subscriber archives the last message". An analyst can clarify these situations with further examination and analysis of the scenarios, once the sharing has been identified. The scenario context relations described below in Section 3.5 provide a means of recording the results of the analysis for later reuse.

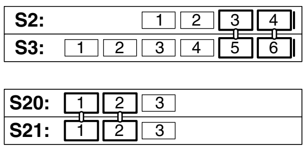


Fig. 9. Two scenarios sharing a suffix (EMS scenarios S2 and S3), and two sharing a prefix (S20 and S21).

3.5. Context

Each scenario describes activities and behaviors appropriate in some context. The context is created by the events of other scenarios, by events not described, or some combination. Whenever one scenario is intended to create a context for another, or can occur in the same context as another, the relationship should be made evident. There are several approaches for doing this.

3.5.1. Overlaps

One approach, informal and often encountered, is to indicate context by *sharing suffixes and prefixes* among scenarios. The last few events of a scenario S_1 are repeated as the first few events of scenario S_2, and this is used to mean that S_2 may overlap and follow S_1, or more exactly, the behavior described by S_2 (minus its prefix) can occur after the behavior described by S_1. The shared suffix and prefix become more than a description of some behavior, in effect also identifying a specific context. While these overlaps are convenient and intuitive, they cannot indicate whether any other suffixes produce the same context, and if so which ones. The overlap-as-context approach is unable to show anything more than the simplest context relations.

3.5.2. Pre- and postconditions

A second and very common approach is *pre- and postconditions* for scenarios. The state of the scenario's world is abstracted into predicates and variables, and these are used to make logical formulae about that world. The precondition of a scenario is a formula specifying the context that the scenario needs – if the precondition is true, then the scenario may occur – and the postcondition is a formula that will be true in the context the scenario produces. Pre- and postconditions take advantage of logic's expressive power. One scenario produces the context a second scenario needs if the first scenario's postcondition implies the second scenario's precondition. Two contexts are equivalent if the conditions describe them are logically equivalent. The change a scenario produces in the world is reflected in the difference between its pre- and postcondition. Pre- and postconditions provide a much more flexible, expressive, and powerful approach to specifying context.

But pre- and postconditions can be difficult to use effectively. It is challenging to write conditions that force the scenarios to follow each other in exactly the desired relationships. Our experience (confirmed in validation studies) is that the pre- and postconditions given to scenarios very frequently allow scenarios to follow each other in unintended ways, and sometimes prevent scenarios from following each other in intended ways. Analysts use the conditions to communicate and emphasize what they believe to be the most important factors in the context. These factors tend to become obscured when analysts strengthen the conditions enough to force the intended scenario sequences. The conflict tends to worsen as the scenarios and their conditions evolve.

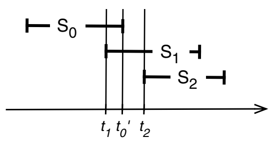


Fig. 10. Conditions of overlapping scenarios S_1, S_2, and S_3.

A second limitation on pre- and postconditions' usefulness is that strictly speaking they only describe the context at the beginning and end of each scenario. Scenarios often overlap, either by sharing a suffix and prefix as described earlier or more generally by describing behaviors that could overlap in time. The contexts at the beginnings and endings of overlapping scenarios do not show how the scenarios can occur, because those contexts are not simultaneous. Fig. 10 shows three scenarios overlapping in time. The context specified by S_0's postcondition occurs after S_1 begins and before S_2. Even if the preconditions of S_1 and S_2 are identical with S_1's postcondition, without more information it is impossible to say whether or not the figure illustrates a possible or desired behavior. We do not have enough information to say whether the context is already established by t_1, or persists until t_2, S_0 and S_1 could change the context over this interval, as could other scenarios not shown, or activities not covered by any scenario.

A third limitation is that pre- and postconditions on the state of the world do not distinguish other context relations such as mutual exclusion between two alternative scenarios that can occur in the same context.

We have not seen any of these issues addressed effectively in ordinary requirements practice. The commonly seen approach is to simply to ignore these issues, rely on intuition and verbal agreements in determining the contexts for each scenario and identifying any problems involving them, and postpone final identification and resolution of conflicts and misunderstandings until later in the development process.

3.5.3. Scenario context diagrams

A step towards one possible informal approach is illustrated in Fig. 11. This diagram describes the contexts needed by and produced by EMS scenario S2 "Subscriber *s* authentication", in terms of the other EMS scenarios. Here we see the scenario that creates a context in which S2 can take place (S0 "EMS Startup"), and the scenarios that make use of the context that S2 creates (S12 "Subscriber *s* listens to a message" and S30 "Subscriber *s* disconnects from EMS"). There are also indications of which scenarios may produce contexts that prevent S2 from taking place, and others that S2 in turn prevents; S1 "EMS Shutdown" prevents S2 from beginning, as does S2 itself, while S3 "Subscriber *s* authentication from unsubscribed phone" prevents S2 from completing (at event 4), in our analysis, and is itself prevented by S2. The 'allow' and 'prevent' arrows may have cardinalities: for example, an occurrence of S0 allows many occurrences (*) of S2, while each occurrence of S2 only allows a single occurrence (1) of S30.

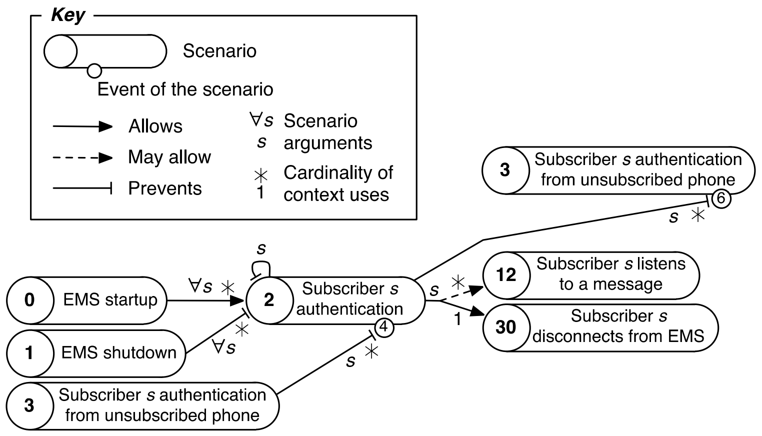

Key
Scenario
Event of the scenario
Allows
May allow
Prevents
∀s s Scenario arguments
Cardinality of context uses
0 EMS startup
1 EMS shutdown
2 Subscriber s authentication
3 Subscriber s authentication from unsubscribed phone
12 Subscriber s listens to a message
30 Subscriber s disconnects from EMS
∀s
s
4
6
1

Fig. 11. Context diagram for EMS scenario S0 "EMS startup".

While the information recorded in a scenario context diagram is useful as a reference, perhaps the greater benefit of the diagrams is that they direct the analyst's attention to contextual connections between scenarios that otherwise may be overlooked. In working through the relations between scenarios one of which produces context another needs, and between scenarios that may or may not produce or need equivalent contexts, omissions and misunderstandings are brought to the surface that may otherwise remain unexamined until later when they may be more expensive or difficult to address.

We note that scenario context diagrams often require that the scenarios involved be explicitly parameterized in order to express useful relationships. In this example, the subscriber s is assumed as a parameter to most of the scenarios, so that when a particular subscriber s_1 authenticates in S2, only that same subscriber s_1 is thereby allowed to disconnect in S30. At the time of this research, we did not have the technical foundation (in terms of a suitable scenario languages and tools) that would support explicit parameterization.

We also note that there is often not enough information available at the requirements stage to make a complete context diagram. It appears that context diagrams are by necessity an informal adjunct to scenarios, useful to record such relationships as can be inferred and as a visual aid for discussion analysis. Within these limitations, they can be useful for exploring and recording analyst's understanding of each scenario's context, and in the incomplete and evolving state of knowledge that requirements analysts work with, can be more effective than event overlaps or pre- and postconditions.

3.6. Similarity

Duplicate scenarios can occur when an analyst writing a scenario does not realize that the collection already contains a scenario describing the same behavior; or when during scenario evolution two scenarios converge to describe the same behavior, although they initially described two distinct behaviors. Exact duplication of a scenario in a collection adds unnecessary work for maintaining both scenarios, and the risk that one of the duplicates will be updated and the other will not, making the collection ambiguous and internally inconsistent; near-duplication of a scenario in a collection introduces ambiguity and inconsistency directly. In a large collection, duplication can go unnoticed if it must be detected manually. An automated similarity measure focuses an analyst's attention on the scenarios most likely to be exact or near duplicates, so that fewer scenarios need be examined to eliminate duplication. A similarity measure can also help an analyst identify related scenarios that must change together, and to search for scenarios with certain attributes and events.

A *similarity measure* is a function that produces a number expressing the degree of similarity between two scenarios. Although no other researchers have published specifically on similarity measures between scenarios, there has been research on measuring similarity between various kinds of entities [22,50]. Here, we discuss a similarity measure specifically tailored for scenarios formalized as described in Section 2.2 [6]. To measure similarity, we consider each scenario as a set of attribute values. We assign the attributes embedded in episodes and events to the scenario in which they appear, so that all attributes of a scenario are examined.

Our similarity measure considers only the syntactic representation of scenarios. It examines whether two attribute values are identical, but not any other relation between them (such as ordering or subsumption) that might arise from a semantic structure. This syntactic focus has several advantages: it is easier to understand, the results of the measure map in an intuitive way to the scenarios, and the extra labor of constructing a semantic structure is not needed.

This similarity measure has several important properties. First, its values are normalized to lie between 0 and 1, with 0 indicating complete dissimilarity and 1 indicating equality. Second, it is customizable for different applications, including scenario search mechanisms and grouping strategies. Third, it can be computed using a computer algorithm, so that similarity can be measured without costly and time-consuming human evaluations. Fourth, it uses attribute glossaries, described above, to simplify scenario comparisons.

For scenario comparison, we select scenario attributes with common values stored in a glossary. Then, "equivalent" attributes are those with the same name in the glossary. In a more complicated strategy, an algorithm could classify equivalent attribute values according to an external semantic structure. Here, we limit the attributes considered to those that require the use of glossaries.

The similarity measure compares scenarios as sets (i.e., unordered lists without duplication) of attribute values, and examines these sets for overlap. Other classes of similarity measures account for sequence of elements such as events [42]; our measure does not, in order to produce a measure that is more easily understood. Thus, for two scenarios S_1 and S_2, each has an associated list of attribute values. Consider the lists of attributes for two scenarios S_1 and S_2:

$$S_1 = \{Actor_3, Actor_5, Actor_6, Goal_2, Purpose_2, Viewpoint_1, ConcretenessLevel_0\};$$
$$S_2 = \{Actor_3, Actor_4, Actor_5, Actor_7, Goal_2, Purpose_1, Viewpoint_1, ConcretenessLevel_0\}.$$

We define the *similarity measure* $S(S_1,S_2)$, the similarity between scenarios S_1 and S_2, as the number of common attribute values in each attribute list, divided by the sum of the sizes of each attribute list, (see [62] for other ratio models):

$$S(S_1, S_2) = \frac{2 \times |S_1 \cap S_2|}{(|S_1| + |S_2|)}$$

The factor of two normalizes the result so that identical scenarios have a similarity of 1. Therefore, the similarity between scenarios S_1 and S_2 is $S(S_1, S_2) = 10/15 = 0.667$. The similarity measure can be seen as a percentage of overlap, where the minimal similarity value of 0 indicates no similarity, and the maximal value of 1 indicates complete overlap in the attribute lists.

In this simple scheme, each attribute value can be considered to have a "weight" of 1. A family of similarity measures arises when we allow different weighting schemes, where each attribute or attribute value is assigned a weight between 0 and 1. Then, when the similarity measure is taken, each attribute value in the measure is multiplied by its weight. One weighting function might, for example, assign a weight of 1 to each actor, and a weight of 0 to all other attributes. With this weighting, scenarios S_1 and S_2 would have a similarity measure of 2/7, about 0.289; by this measure, the two scenarios are much less similar. In this way, the weighted similarity measure emphasizes similarity in particular attributes (by assigning them high weights) and ignores difference in others (by assigning them weights of zero). This can be used for grouping similar scenarios based on particular attribute values, or for scenario searches. The weighted similarity measure can be particularly important for episode searching and matching.

Mathematically, we can write the weighted extension of S as SW, the weighted similarity measure between two scenarios, where a denotes an attribute, and $wt(a)$ denotes the weight assigned to attribute a. Note that, to avoid division by zero, we define the similarity between two scenarios to be 0 if all their attribute values have zero weights:

$$SW(S_1, S_2) = \frac{\sum_{a \in S_1 \cap S_2} 2 \cdot wt(a)}{\sum_{a \in S_1} wt(a) + \sum_{a \in S_2} wt(a)}$$

We also extend these measures to groups of any number of scenarios by taking the average of the similarity of each pair of scenarios. We extend S to S^*:

$$S^*(S_1, S_2, \ldots, S_n) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} S(S_i, S_j)}{n(n-1)/2}$$

For example, to calculate the similarity measure for S_1, S_2, and S_3, where S_1 and S_2 are given above, and

$$S_3 = \{Actor_1, Actor_4, Actor_6, Actor_7, Goal_1, Purpose_1, Viewpoint_3, ConcretenessLevel_0\},$$

we must compute

$$S^*(S_1, S_2, S_3) = \frac{(S(S_1, S_2) + S(S_1, S_3) + S(S_2, S_3))}{3} = 0.478$$

Computed in this fashion, S^* gives an average of the similarities of each pair of scenarios in a group.

We may also extend SW to SW^* in a similar fashion:

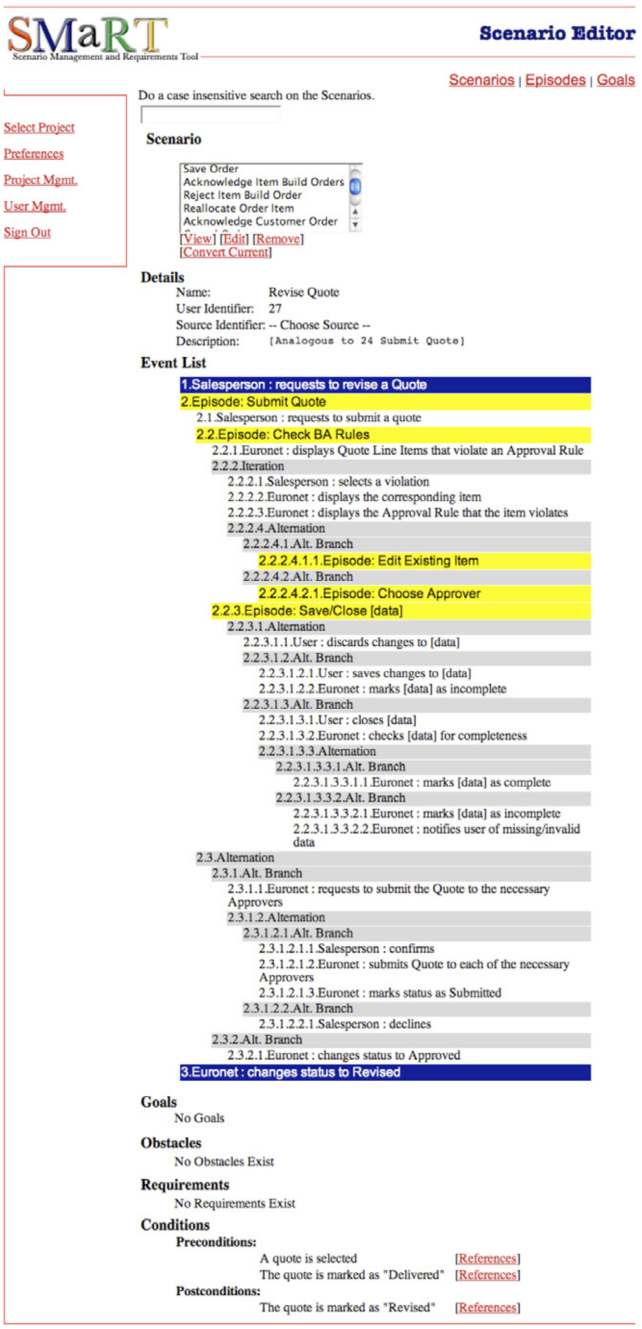
$$SW^*(S_1, S_2, \ldots, S_n) = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} SW(S_i, S_j)}{n(n-1)/2}$$

These two extensions, weighting and group comparisons, build a family of measures which are powerful tools for scenario management. The general measure can be used to search for and select scenarios with particular characteristics from a scenario database, quickly and automatically, as in [44]. It can also help identify accidental duplication in a large set of scenarios. Using the similarity measures, SMaRT will present similar scenarios, or episodes, to an analyst for evaluation and possible elimination. Because of their simplicity and flexibility, the similarity measures provide both quick automatic processing of data that would take hours by hand, and a measure that corresponds to our intuitions about scenario similarities.

4. SMaRT

The Scenario Management and Requirements Tool (SMaRT) developed at NCSU assists analysts working with scenarios by supporting the scenario structure described above (Section 2.2), glossaries of attribute values, episode management, and efficient authoring, navigation, and editing of scenarios.

SMaRT is implemented using a database on a central server and is accessed over the Internet through a browser. Scenarios in this database are organized into projects, and each project has a list of analysts allowed to edit or view the scenarios, episodes, and glossaries for that project. A scenario is created by associating values in the project's glossaries with the attributes for that scenario. New values may be added as needed to the glossaries, existing values can be edited (with the changes appearing everywhere the value is used), and unused values can be found and deleted. The attributes include actors, actions, events, goals, obstacles, requirements, and conditions. SMaRT maintains a clickable cross-reference for each attribute value giving every place where it is used. The events of a scenario can include simple events, episode references, lists of events treated as a single compound event, iteration of events, and alternation among events. Episode references can be expanded to show the events of the episode instead of the name of the episode, and the events of the episode can be edited if desired in this context. Clicking on the name in an episode reference takes the analyst to an episode window showing all the information about that episode.



Do a case insensitive search on the Scenarios.

Scenario

Save Order
Acknowledge Item Build Orders
Reject Item Build Order
Reallocate Order Item
Acknowledge Customer Order

Details

Name:	Revise Quote
User Identifier:	27
Source Identifier:	-- Choose Source --
Description:	[Analogous to 24 Submit Quote]

Event List

```
1.Salesperson : requests to revise a Quote
2.Episode: Submit Quote
  2.1.Salesperson : requests to submit a quote
  2.2.Episode: Check BA Rules
    2.2.1.Euronet : displays Quote Line Items that violate an Approval Rule
    2.2.2.Iteration
      2.2.2.1.Salesperson : selects a violation
      2.2.2.2.Euronet : displays the corresponding item
      2.2.2.3.Euronet : displays the Approval Rule that the item violates
      2.2.2.4.Alternation
        2.2.2.4.1.Alt. Branch
          2.2.2.4.1.1.Episode: Edit Existing Item
        2.2.2.4.2.Alt. Branch
          2.2.2.4.2.1.Episode: Choose Approver
    2.2.3.Episode: Save/Close [data]
      2.2.3.1.Alternation
        2.2.3.1.1.User : discards changes to [data]
        2.2.3.1.2.Alt. Branch
          2.2.3.1.2.1.User : saves changes to [data]
          2.2.3.1.2.2.Euronet : marks [data] as incomplete
        2.2.3.1.3.Alt. Branch
          2.2.3.1.3.1.User : closes [data]
          2.2.3.1.3.2.Euronet : checks [data] for completeness
          2.2.3.1.3.3.Alternation
            2.2.3.1.3.3.1.Alt. Branch
              2.2.3.1.3.3.1.1.Euronet : marks [data] as complete
            2.2.3.1.3.3.2.Alt. Branch
              2.2.3.1.3.3.2.1.Euronet : marks [data] as incomplete
              2.2.3.1.3.3.2.2.Euronet : notifies user of missing/invalid
              data
  2.3.Alternation
    2.3.1.Alt. Branch
      2.3.1.1.Euronet : requests to submit the Quote to the necessary
      Approvers
      2.3.1.2.Alternation
        2.3.1.2.1.Alt. Branch
          2.3.1.2.1.1.Salesperson : confirms
          2.3.1.2.1.2.Euronet : submits Quote to each of the necessary
          Approvers
          2.3.1.2.1.3.Euronet : marks status as Submitted
        2.3.1.2.2.Alt. Branch
          2.3.1.2.2.1.Salesperson : declines
    2.3.2.Alt. Branch
      2.3.2.1.Euronet : changes status to Approved
3.Euronet : changes status to Revised
```

Goals

No Goals

Obstacles

No Obstacles Exist

Requirements

No Requirements Exist

Conditions

Preconditions:

A quote is selected	[References]
The quote is marked as "Delivered"	[References]

Postconditions:

The quote is marked as "Revised"	[References]

Fig. 12. SMaRT Scenario Editor screen showing revised Euronet scenario 27.

Having scenarios expressed in the structure supported by SMaRT gives a basis for analyzing scenarios both individually and as an entire project, and forms a foundation for expanding automated support for working with scenarios and new presentations and analyses of the information contained in scenarios.

A screen shot of the Scenario Editor is shown in Fig. 12. The scenario being edited is revised Euronet scenario 27 ''Revise Quote''.

Future work for SMaRT includes implementation of similarity measures, expanded cross-reference capability, and integration of support for shared event diagrams, episode reference diagrams, and scenario context diagrams.

5. Validation

In this section we summarize the results of two validation studies, the first on EMS (Enhanced Messaging System), a voice mail system used by BellSouth Telecommunications to prototype new features for its products, and the second on Euronet, a quote management system developed and used internally by Asea Brown Boveri (ABB). The EMS study was conducted with a domain expert from BellSouth, so that we were able to engage in dialogue during the course of the study and present some results for discussion and feedback. The Euronet study was done on the ABB requirements specification, but since ABB had completed the Euronet project some time earlier, we were not able to interact significantly with analysts or stakeholders there or present our results for feedback. In both cases we conducted the work introspectively and kept detailed logs so that we could go back and analyze our conclusions and the evidence behind them.

5.1. Evaluation criteria for the studies

We consider our approach to be a basis for automating scenario tasks that are difficult, time-consuming, or impractical to do by hand, and as a result a basis for producing scenarios of higher quality (better-defined, more coverage, more nearly minimal, and more coherent) that better support efficient development of appropriate systems. Therefore, we can evaluate the approach by showing whether it achieves these qualities as well or better than working by hand, but significantly faster or easier due to automation; or that it provides additional support or analyses impractical to do by hand that can be used to improve scenario quality; or that it supports greater expressiveness that can be used to produce higher-quality scenario collections.

We show this by comparing scenarios produced or evolved using our approach, to scenarios produced by other people and/or using other methods. Where possible, we compare the scenarios based on objective characteristics such as the number and scope of problems that were found, and the person-hours that were necessary to find them. Where that is not possible, we compare the scenarios based on our judgement and on our experience in both the research world and the industrial software development world. For the EMS study, we were able to get some independent feedback from the stakeholder.

In addition, for the Euronet study, we can evaluate both the effectiveness and the efficiency of our approach in terms of quality results achieved and time required to achieve them, by comparison with the results from and time required for the earlier goal-based study of Euronet [13].

5.2. The EMS study

EMS is a comprehensive telephone voice messaging system used by BellSouth Telecommunications to prototype new features for its products. It supports a wide range of functionality, including: access and authentication; subscriber interactions with the EMS (e.g., notifications and message processing); caller interactions with the EMS (e.g., recording of incoming messages and the marking of certain messages as urgent); and subscriber configuration and management functions (e.g., recording announcements and archiving messages).

We performed the study described here during the second half of the collection and specification of the EMS requirements for another research project that needed a clear requirements specification as the basis for an evaluation of later-phase development methodologies.

EMS was a good basis for a validation study of scenario support for effective requirements for a number of reasons. It was a real system, of manageable size, whose requirements and scenarios were not determined by us [52]. A domain expert from BellSouth Telecommunications controlled the content of the requirements and scenarios, limiting the effect of any biases on the part of the authors and ensuring that the scenario collection was substantial and genuine. Its requirements and scenarios were written by three experienced and capable requirements engineers (the domain expert and the two authors), who conducted inspections, detailed walkthroughs, and reviews over a period of eight weeks before the study was begun. The final scenarios were used successfully as the requirements for developing a prototype system as part of another research project.

For this study, we started with the requirements and scenarios as they stood after the eight weeks of inspections, walkthroughs, and reviews, and our notes and logs of that process. We explored what analyses, indexes, cross-references, and diagrams would be useful in producing effective requirements, especially those that could in principle be produced automatically. We manually produced examples of them in order to evaluate their effectiveness, and examined in what ways and to what extent they aided us in improving the well-definedness, coverage, minimality, and coherence of the EMS requirements (including the scenarios). The scenarios already made use of a glossary and episodes at the beginning of this study. We analyzed, extended, and improved the glossary and made preliminary versions of attribute glossaries. We analyzed the episodes (originally there were two, later only one) by expanding them inline in each scenario sharing them, to verify that the episode relationship was appropriate and to explore analogous conditions across all the scenarios sharing an episode. We identified similar or identical events shared by two or more scenarios, manually constructed shared event diagrams, and analyzed the consequences of the sharing for the scenarios. We explored the allowed and desired contexts for the scenarios, individually and in terms of other scenarios, and examined alternate ways of expressing and working with the contexts, including a predecessor of scenario context diagrams. Finally, we manually calculated similarity among a subset of the scenarios and compared the results with our intuitive perceptions of their similarity, and explored how the similarity could be used in increasing the quality of the requirements. We continued keeping detailed notes and logs during the study, and later were able to analyze them in some detail to determine what had happened and why.

A preliminary version of the EMS study was presented in an earlier publication [4].

5.2.1. Study artifacts

The EMS specification consists of requirements, scenarios, episodes, and glossaries for terms and conditions. Table 1 gives the total of each kind before and after the study. Fig. 13 presents examples from after the study, and the entire collection is available online [11].

The requirements were expressed in traditional prose form, and used the words and phrases defined in the glossary. For example, R3.2.13 specifies that a subscriber can

Table 1
Numbers of EMS artifacts, before and after the study

	Before	After
Requirements	45	44
Scenarios	32	40
Episodes	2	1
Defined terms	17	21
Conditions	10	14

A *recipient* is either a single phone number, or a group of phone numbers already defined by a subscriber and identified by a two-digit number.
The *subscriber* is a person for whom EMS provides a voice mailbox.

R3.2.13.	EMS shall allow a *subscriber* to record a message and send it to a *recipient*. This can occur at any point, and the *recipient* need not be a caller who has left a message.
R4.7	EMS shall allow a caller to leave a message for a *subscriber* in two ways: by calling the *subscriber*'s number (if the *subscriber* doesn't answer), and by calling EMS directly.

Authenticated: The subscriber is connected to EMS and is authenticated.

S26. Subscriber records a message and sends it to someone

Requirements:	R3.2.13.
Precondition:	Authenticated.
Postcondition:	Authenticated.

1.	The subscriber dials the "record message" command.
2.	*Iteration with explicit exit:*
2.1.	EMS asks for the telephone phone number to which the message will be sent.
2.2.	The subscriber dials the telephone number, followed by a non-numeric number-ending command (such as '#').
2.3.	EMS says the telephone number and asks the subscriber to confirm or reject it.
2.4.	*Alternation:*
2.4.1.	The subscriber dials the confirmation command. Exit from iteration
2.4.2.	The subscriber dials the rejection command.
3.	Episode: MakeRecording
4.	EMS sends the recorded message to the telephone number.

Fig. 13. Example EMS definitions, requirements, state variable and scenario.

record a message and send it to a recipient, and the glossary defines what "recipient" means in this context.

The scenarios ranged in length from 1 to 12 simple events. Each scenario was traced back to up to four requirements; the example scenario S26 is traced back to requirement R3.2.13. Each scenario had up to 4 preconditions and postconditions; the pre- and postcondition for S26 are both the primitive condition "Authenticated". Episodes (identified manually) were defined and were referred to by three or four scenarios. S26's event 3 is a reference to the episode "MakeRecording" (not shown here). S26's event sequence contains an iteration (event 2), whose effect is to repeat subevents 2.1, 2.2, 2.3, and 2.4 until event 2.4.1 signals termination of the iteration. Event 2.4 is also a compound event, an alternation among choices 2.4.1 and 2.4.2. Iteration and alternation are well-known concepts, but we note that they are not commonly used for scenarios or use cases in practice.

Several "menu trees" (one is shown in Fig. 14) intended to express the possible sequences of scenarios were produced for the scenarios at the direction of the domain expert. These trees were based, we believe, on the fact that all EMS activity was initiated through a hierarchical voice menu presented to EMS users. The nodes of the trees were scenarios, and the indicated sequences were those that moved between adjacent scenarios in the tree, beginning at the root and continuing down or up at each node. However, there appeared to the authors to be some question whether or not sibling nodes were consistently considered adjacent in this sense.

1. Configuration
   (a) Change passcode.
   (b) Configure announcement.
   (c) Set up group of phone numbers.
   (d) Up to parent menu.
2. Check for new messages.
3. Process messages
   (a) Play the first message.
      i. Skip around in the current message.
   (b) Play the next message (a.k.a. "skip").
      i. Skip around in the current message.
   (c) Play the time the current message was received.
   (d) Erase the current message, and move to the next one.
   (e) Archive the current (new or held) message, and move the next one
   (f) Reply to the current message.
   (g) Forward the current message.
   (h) Forward the current message with a preface.
   (i) Call the person who left the current message.
   (j) Record a message and send it to a phone number.
   (k) Up to parent menu.

Fig. 14. Subscriber menu tree.

5.2.2. Lessons learned

5.2.2.1. Glossaries help establish a common understanding of terms and concepts. Our experience confirmed what others have found [36,63]. The glossaries of terms and primitive conditions were invaluable for helping us record our gradually-increasing domain knowledge and helping us consolidate our understanding of the problem and of the intended solution. An example is the group of terms that define the various states a message passes through. Over the course of the study, the successive versions of the glossary show how we extended our knowledge beginning with messages that could be *new*, *archived*, *old*, or *urgent*, extending later to *held* and *erased*, and eventually restricting *old* to be either *old archived* or *old held*, and separating *urgent* into an orthogonal category that finally included *private* as well. The defined terms formed the scaffolding on which we then wrote scenarios exploring and then specifying what could happen with messages of in the various states.

We found we were unable to create and use attribute-value glossaries effectively without tool support; we explored this technique further with SMaRT and the Euronet study.

5.2.2.2. Scenario context diagrams help uncover missing scenarios and requirements. The process of constructing scenario context diagrams requires that analysts walk through paths spanning several scenarios, comparing pairs of scenarios for degrees of equivalence and comparing the contexts required and produced by the various scenarios. Considering the scenarios in this way draws more focused attention to these relationships than reviews or walkthroughs, and some omissions or misunderstandings that had previously slipped past are caught.

The following missing scenarios were identified almost immediately during construction of the EMS scenario context diagrams:

- S0 "EMS startup" was discovered because there was no scenario to provide initial context for any chain of the others.
- S1 "EMS shutdown", S29 "Subscriber disconnects from EMS", and S39 "Caller disconnects from EMS" were discovered because there were no scenarios to terminate the contexts in which others could take place.
- S13 "Subscriber has no more messages to listen to" was discovered because S12 "Subscriber listens to a message" could produce a context in which no more messages were available, but no scenarios existed to make use of it.S15 "Subscriber would skip to next message but has no more messages", S19 "Subscriber erases the last message", and S21 "Subscriber archives the last message" were discovered for analogous reasons.
- S38 "Caller calls EMS directly and leaves a message" (Fig. 1) and the corresponding requirement R4.7 (Fig. 13) were discovered to be missing during the process of matching up scenarios using or making equivalent contexts. This important requirement had been discussed early in the initial eight weeks but never specified, and weeks of reviews and walkthroughs had failed to uncover the omission.S38 was discovered by visually comparing the diagram configuration for subscriber scenarios to that for caller scenarios; the presence of the corresponding scenario for subscribers, S26 (Fig. 13), was a prominent difference.

5.2.2.3. Scenario context diagrams help uncover requirements errors. The same diagram walkthroughs and comparisons that draw focused attention to scenarios also focus attention on requirements. They force analysts to think about individual requirements concretely and in specific contexts. As with scenarios, this focused attention identifies errors that had slipped past reviews and single-scenario walkthroughs.

We discovered one such requirements error by noting scenarios that lead somewhere but have nowhere to come from, or vice versa. Initial and terminal scenarios like S0 "EMS startup" and S1 "EMS shutdown" do this intentionally, but other scenarios do not, and the presence of other scenarios that either produce a context no other scenario can use, or need a context no other scenario produces, is a sign that something is not right. In this case, we observed that the pre-study scenario "Subscriber listens to an archived message" (which does not exist in the post-study scenarios [11]) lacked appropriate pre- and postconditions to describe its initial and final contexts. While tracing what these conditions should be, we discovered that we had incorrectly separated this scenario from "Subscriber listens to a new or held message" (which also does not exist post-study). These two behaviors were originally to have been distinct and the two scenarios and two corresponding requirements recorded this. Post-study analysis of our logs showed that subsequent discussions with the domain expert had indicated that the two behaviors should instead be the same, but this change was somehow not recorded in either scenarios or requirements, and the error was missed by all intervening reviews and walkthroughs.

A second requirements error was discovered while tracing these two scenarios back to requirements. We found that not only did the two corresponding requirements incorrectly separate those two cases, but that a third requirement both scenarios traced back to was also incorrect. That was the presentation sequence requirement R.3.2.1, which incorrectly specified that messages be presented oldest-to-newest, rather than newest-to-oldest. This mistake had also been missed despite reviews and walkthroughs by the domain expert and the two authors. The two versions of requirement R3.2.1 are shown in Table 2.

5.2.2.4. Scenario context diagrams express when each scenario can occur. Scenario context diagrams makes explicit the temporal relationships between scenarios. Before we constructed the diagrams, we had a certain degree of intuition and informal knowledge of the patterns in which the EMS scenarios could occur, and had assigned pre- and postconditions to the scenarios that suggested the appropriate contexts but that were not strictly correct and often not helpful. Drawing diagrams of scenario context made our intuitive knowledge explicit, revealed ways in which our intuitive and informal knowledge had been inadequate or incorrect, and provided a direct and more accurate expression of what the pre- and postconditions had attempted to express indirectly.

5.2.2.5. Similarity measures can identify similar scenarios. Although the results were not conclusive, we found evidence that our similarity measure identified scenarios that we independently judged as similar [7,23]. We validated the similarity measure on a group of 12 EMS scenarios. There were 66 possible pairs of the 12 scenarios ($12 \times 11/2$). Of these 66 pairs, we judged 27 pairs to be similar scenarios, and the remainder to be dissimilar. The necessary calculations for the similarity measure were done using a spreadsheet to record the necessary data and calculate the results; setting up the calculations and checking them required a week of full-time work. It is clear that manually calculating similarity with our measure is time-consuming,

Table 2
EMS requirement R3.2.1 and some related terms

An *old archived message* is an archived message that was received more than some certain interval of time ago.
An *urgent* message is one that the caller that left it identified as such by a command; urgent messages are treated with a higher priority in certain cases.

Pre-study R3.2.1:	EMS shall present messages in chronological order (oldest to newest), except that EMS shall present all *urgent* new messages to the subscriber before presenting any new message that is not urgent.
Post-study R3.2.1:	EMS shall present messages in the following order: *old archived messages* and old held messages; then *urgent* new messages, from newest to oldest; and finally all other messages, from newest to oldest.

subject to human error, and impractical – tool support is essential. The calculation gave all attributes equal weight and treated each episode reference as a single attribute (rather than expanding episode references and using the subevent actors and actions as individual attributes).

Examination of the results shows some promise for using automated similarity calculations. We found that of the 66 similarity values, the lowest 32 were for pairs judged dissimilar; 19 of the next 22 highest were for pairs judged similar; and the 12 highest values were for pairs judged similar. The similarity values are presented in Table 3. These results support our hypothesis that the measure can be used to direct analysts' attention to similar scenarios that require further attention, although the measure is not a substitute for additional human judgement.

5.2.3. Stakeholder feedback

For the EMS study, we had access to a stakeholder, namely the domain expert from BellSouth who was our customer for the other EMS research project. Our interactions with him consisted of one face-to-face meeting and conference calls approximately every two weeks over a 12-week period. During that time, we elicited the EMS requirements and scenarios from him and analyzed and revised them in successive review cycles, until they were acceptable to him and we felt we understood them and were all in agreement.

The authors emailed the stakeholder successive revisions of the requirements and scenarios, and during the calls we conducted reviews and walkthroughs and discussed issues raised during them or by any of the participants. The stakeholder was pleased that we were able to identify problems and improve the specification, no matter what techniques we were using. For example, the identification of S38 "Caller calls EMS directly and leaves a message" and the corresponding requirement R4.7 were of considerable interest; but the fact that we uncovered them by analyzing scenario contexts was not significant to him.

The stakeholder also had little interest in a number of issues that we felt were problems. Examples were the initial and terminal scenarios for EMS as a whole, our analysis of the menu trees in terms of scenario context diagrams, and some clarifications of the definitions of defined terms. All these were cases in which we did not understand the requirements specification, even though the stakeholder did and was satisfied with it. As in the case of problems of interest to the stakeholder, we found term glossaries, shared events, and scenario context diagrams to be effective and helpful in identifying problems we felt needed further attention and in resolving them successfully. This efficiency and effectiveness was of value even for problems the stakeholder did not find important, because analysts need to understand requirements just as stakeholders do.

5.2.4. Discussion

Our experience in the EMS study confirmed that term glossaries support scenario well-definedness, and contribute to coherence and minimality, as others have noted also.

The EMS scenarios did not make much use of episodes, possibly because users of the system send input through a keypad and receive output as spoken text so that the fundamental design of the system focuses on a relatively small number of distinct behaviors. In addition, the lack of automated support for episodes meant all the analyses and conversions had to be done manually, so that we did fewer of them. Consequently there was little opportunity in this study to explore the effect of episodes and episode reference diagrams on the quality of scenarios as requirements.

We found that scenario context diagrams helped indicate incoherence and gaps in coverage, and direct attention to the areas that would resolve it. Identifying these gaps and areas of incoherence is the necessary first step towards improving the coverage and coherence of the scenario collection. We distinguished three kinds of gaps in this study, based on how instances of them were identified. The easiest kind to find, and the kind that is probably of the least interest, was missing behavior consisting of initial and terminal scenarios such as S0, "EMS startup". The second kind was missing behavior that was turned up through the analysis of pre- and postconditions for the relationships required by the scenario context diagrams, such as S13, "Subscriber has no more messages to listen to". The third and most interesting kind was behavior whose absence was indicated visually by analogy between the scenario context diagrams, exemplified by S38, "Caller calls EMS directly and leaves a message".

Some of the temporal and causal relationships among scenarios were indicated by the "menu trees" of scenarios. Other relationships were expressed, not very effectively, by the scenario pre- and postconditions. The scenario context diagrams expressed these relationships more fully and more effectively. We also found that the scenario context dia-

Table 3
Scenario pairs, sorted from least to greatest calculated similarity

Rank	*A*	*B*	*S*(*A*,*B*)	Rank	*A*	*B*	*S*(*A*,*B*)
1.	11	33	0.07	34.	**26**	**31**	**0.21**
2.	12	33	0.08	35.	**31**	**32**	**0.21**
3.	11	26	0.08	36.	**1**	**11**	**0.25**
4.	11	32	0.08	37.	28	31	0.25
5.	12	26	0.09	38.	28	33	0.25
6.	12	32	0.09	39.	**1**	**12**	**0.26**
7.	23	33	0.09	40.	**27**	**33**	**0.27**
8.	1	28	0.10	41.	**29**	**33**	**0.27**
9.	11	28	0.10	42.	**30**	**33**	**0.27**
10.	12	28	0.10	43.	**27**	**31**	**0.29**
11.	23	26	0.10	44.	**29**	**31**	**0.29**
12.	23	32	0.10	45.	**30**	**31**	**0.29**
13.	1	27	0.11	46.	**1**	**23**	**0.30**
14.	1	29	0.11	47.	**26**	**27**	**0.32**
15.	1	30	0.11	48.	**26**	**29**	**0.32**
16.	1	31	0.11	49.	**26**	**30**	**0.32**
17.	11	27	0.11	50.	**27**	**32**	**0.32**
18.	11	29	0.11	51.	**29**	**32**	**0.32**
19.	11	30	0.11	52.	**30**	**32**	**0.32**
20.	11	31	0.11	53.	**26**	**28**	**0.38**
21.	12	27	0.11	54.	28	32	0.38
22.	12	29	0.11	55.	**11**	**23**	**0.40**
23.	12	30	0.11	56.	**12**	**23**	**0.42**
24.	12	31	0.11	57.	**32**	**33**	**0.44**
25.	23	28	0.12	58.	**28**	**29**	**0.50**
26.	23	27	0.13	59.	**28**	**30**	**0.50**
27.	23	29	0.13	60.	**11**	**12**	**0.52**
28.	23	30	0.13	61.	**27**	**29**	**0.57**
29.	23	31	0.13	62.	**27**	**30**	**0.57**
30.	1	33	0.15	63.	**26**	**32**	**0.58**
31.	1	26	0.17	64.	**29**	**30**	**0.71**
32.	1	32	0.17	65.	**26**	**33**	**0.74**
33.	**31**	**33**	**0.18**	66.	**27**	**28**	**0.88**

Similarity was calculated including actions and after reconciliation of synonyms. The pairs in boldface are those deemed intuitively similar by the analyst.

grams specified context relationships that we had been informally aware of but which did not fit into the format of the "menu tree". Scenario context diagrams thus allow us to improve the coverage and coherence of the scenario collection by expressing significant relationships that are not otherwise evident.

We found that calculations of similarity were impractical without automated support, and that there was promise that the calculations could help support human intuition once automated support is available.

5.3. The Euronet study

Euronet is a quote management system used internally by ABB. It supports salespeople, design engineers, plant managers, and business analysts as they create and manage quotes to customers, the line items that give the details of each quote, customer orders, the item build orders that give the details of each order, and related information. The initial development of Euronet was not successful, attributed by ABB to poor quality of the requirements specification. This initial requirements specification, consisting primarily of 52 use cases, was the subject of a research study applying a goal-based analysis to the use cases using GBRAM (Goal-Based Requirements Analysis Method) [13]. The requirements specification was then revised and the system was successfully implemented; we did not have access to the revised requirements, or any significant interaction with stakeholders. ABB graciously allowed the use of the initial requirements specification as the basis for this research.

The Euronet requirements specification was a good subject for a study for a number of reasons. It was an industrial specification for a real system more complex than EMS but still of manageable size, and specified with no influence from the authors. Its level of quality was interesting, as it had been initially judged to be of sufficient quality to serve as a basis for system development, but later its quality had been found to be unsatisfactory. The Euronet requirements specification had been studied in the earlier goal analysis, and we had access to its data. This gave us two points of comparison for our own study: the original specification, and the results from the goal analysis.

For this study we used SMaRT, which was being developed as the study was conducted. We entered the 52 use cases into SMaRT and used its automated analyses, supplemented by manual analyses that could be automated or were planned for later implementation, to understand and analyze the Euronet requirements. Since SMaRT is a scenario tool, we entered each use case's main scenario and other information as either a scenario or an episode. Each event was divided into an actor and an action; use case events that described two or more separate actions were subdivided into individual scenario events so that each could be a single actor and a single action. SMaRT maintained glossaries of actors and of actions so that actors and actions could be reused, thus enhancing minimality but requiring that each actor or action be examined and considered carefully in order to decide if it was equivalent to one already in the glossary, or a new one. We also entered each use case's title, overview, preconditions, and postconditions. We created a glossary of specialized terms and GUI screens used in the use cases. The remaining use case information (notes, secondary scenarios, and revision history) was not used in the study. The "Utilizes" lists of use cases used as episodes by each scenario were not needed, as SMaRT generated this list automatically. We then examined the problems that this process and the analyses turned up, and resolved them to the extent possible. Because we had no access to Euronet stakeholders or developers, we were unable to resolve some problems and questions completely; for example, some of the episodes referred to in the use cases but not defined were impossible for us to define with any confidence. In such cases we noted that the item was undefined or incompletely defined. Scenario similarity calculations were not implemented by SMaRT at that time, and we did not perform manual similarity calculations for Euronet. We manually constructed the episode reference diagram at several points in the study, using the episode index generated by SMaRT. Construction of this diagram is automatable but was not implemented at the time of the study. This diagram was particularly informative for Euronet due to the complex episode relationship among its scenarios.

The study spanned a period of 20 days.

5.3.1. Study artifacts

The Euronet requirements specification consisted of use cases, screen sketches, and a general overview [18]. It included no requirements *per se*. Many of the use cases use others as episodes; some of the episodes are not defined. The screen sketches are not essential to understanding the use cases, and were not used in the study. Table 4 gives the total of each kind before and after the study.

Table 4
Numbers of Euronet artifacts, before and after the study

Before		After	
Use cases not used as episodes	25	Scenarios	32
Use cases used as episodes	27		
Episodes used but not defined	12	Episodes	34
Screen sketches	26		

The use cases are presented in a relatively consistent format. An example use case (UC 27 "Revise Quote") is shown in Fig. 15 (the post-study scenario 27 is shown in Fig. 12). Each of the use cases is numbered (1 through 52) and has a short title. The content of each use case is organized as shown in Table 5. The event lists range in length from 1 to 19 events, with 3 events being most common. The use cases vary from one to six pages in length, with two pages being the most common.

5.3.2. Lessons learned

5.3.2.1. Glossaries and cross-references support well-definedness. Expressing the 52 Euronet use cases with attribute-value pairs and glossaries in SMaRT immediately revealed a large number of problems, ranging from the annoying to the serious. The use cases named in the "Utilizes" section of each use case description frequently do not match the use cases that appear in the event list (for example, in Fig. 15 use case 27's "Utilizes" section lists "B. Use Choose Approver", and no use case named "Choose Approver" is used in its events). The use cases refer to a large number of statuses of various items, but not all these statuses are defined. Ten screens named in the use cases are not defined with a screen shot like the others; four of the ten are probably equivalent to similarly-named defined screens, leaving six that are definitely

Use Case # 27: Revise Quote
Overview:
Any revisions to a Quote that has been marked as "Delivered" are stored and the status is marked as "Revised." e.g., the Salesperson may revise the quote to meet new customer specifications.
Preconditions:
1. Euronet DB contains a complete Quote entry with a status of "Delivered".
2. Shopping Cart Screen is displayed on Salesperson's PC and Salesperson is viewing Quote to be revised.

Main Scenario:
1. Salesperson updates existing quote as required.
   A. Use Edit Quote.
2. Salesperson selects Revise function
3. Euronet checks the revisions for any new violations of BA Rules.
   A. Use Submit Quote.
4. If new violations have been introduced, Euronet requests approval
   A. Use Approve Quote
5. If no new violations are found User assembles a revised quotation package
   A. Use Assemble Quote Package
6. Euronet changes Quote status to "Revised" and returns to Euronet Main Screen

Scenario Notes: *[none]*
Post Conditions:
1. Status of Quote has been changed to "Revised."
2. Euronet Main Screen is displayed on Salesperson's PC.

Required GUI:
A. Shopping Cart Screen
B. Euronet Main Screen
Secondary Scenarios:
1. Salesperson selects Main Screen function at any time: Euronet asks if Salesperson wants to save changes. If Salesperson selects Yes: Euronet stores any changes and returns to Euronet Main Screen. If Salesperson selects No: Euronet returns to Euronet Main Screen.

Uses/Extends:
A. Use Edit Quote
B. Use Choose Approver
Other Requirements: *[none]*
Revision History:
1. Initial Document Release
   A. Date: 11-March-1999
   B. Description: Initial release
   C. Name: *[author]*

Fig. 15. Euronet use case 27 "Revise Quote".

Table 5
Organization of Euronet use cases

Heading	Contents	Occur in
Overview	A summary consisting of 1–3 sentences	All 52
Preconditions	List of 0–2 prose preconditions, usually numbered	51
Main scenario	List of 1–19 prose events, usually numbered;	All 52
	some refer to (‘Use’) 0–7 other use cases	27
Scenario notes	List of 0–2 numbered sentences	4
Postconditions	List of 0–3 prose postconditions, usually numbered	43
Required GUI	List of 0–4 GUI screens for the use case; often does not match screens actually referenced	38
Secondary scenarios	Numbered list of 0–3 alternative scenarios; each is given as a prose sentence or paragraph.	27
Utilizes	List of 0–7 use cases used by this use case; often does not match use cases actually referenced	19
Extends	(*No Euronet use case extends another one*)	
Other requirements	(*No Euronet use case references any*)	
Revision history	Date (11 August 1999) and “Initial release”	

undefined. Of the 26 defined screens, three are never referenced in any use case. Such lacks indicate that the use case collection is not well-defined.

SMaRT automatically generates cross-references for episodes, and cross-references for all glossary values are planned. These cross-references support analyses and simple checks that improve well-definedness but need automated support in order to be practical. For example, in many cases an entity’s status was set but never examined, or examined but never set; this is immediately obvious from the cross-references. We believed many of these inconsistencies were present simply because no analyst’s attention had been drawn to them before.

5.3.2.2. Glossaries help reduce undesirable ambiguity in the specification. The original Euronet specification lacked a term glossary, and we found it difficult to understand the significance of certain terms in the use cases. For example, the verb “close” was clearly used with one or more special meanings which we recognized but did not understand. We also found it difficult to separate out the appropriate meanings of terms that appeared to be used in two senses. For example, “order” was used to mean an order from a customer to ABB, and also an order from ABB to a supplier, and the distinction was not always clear from context. Finally, we found it difficult to tell when two terms were being used for the same concept (for example, it was unclear at first whether “Planner” and “BA Planner” were synonymous). We would have required interaction with stakeholders and analysts to resolve these issues. Even so, it was clear that the use of term glossaries would improve the well-definedness, minimality, and coherence of the specification, and was a necessary prerequisite to identifying and resolving gaps in coverage.

5.3.2.3. Standardizing on a relatively small number of important words makes it easier to find the right action and to express the actions effectively. The events in the original Euronet use cases were worded in no consistent fashion. We found that choosing a set of standard words to use wherever possible greatly reduced the number of separate actions, and at the same time made the events clearer and easier to understand relative to each other. This improved the well-definedness, minimality, and coherence of the scenario collection. Examples were the use of “select” in place of “choose”, “find”, “click”, etc. whenever it was appropriate. This lesson was analogous to the similar finding for goal wordings by Anton et al. [13].

5.3.2.4. Episode reference diagrams support well-definedness and coherence. The episode reference diagram for the use cases highlights the presence of use cases that are referenced but not defined. Twelve apparent use case names are referred to in events but not defined; this indicates a potentially substantial gap in coverage. For example, use case 11 “Edit Quote” (Fig. 16) refers to the undefined episode “Edit Quote Header” in event 3.

Episode reference diagrams also support coherence in the use of individual episodes. They conveniently show all the locations where an episode is used, and for two or more episodes at a time, show all the locations where both (or all) are used together, so that an analyst can compare the various uses to verify that episodes are used comparably and consistently in all locations, either individually or in groups.

5.3.2.5. Episode reference diagrams visually indicates likely problem areas. We found two kinds of problem indications: deep paths and clusters.

The episode reference diagram (Fig. 6) has two top-to-bottom paths of 5 use cases each (for example, 27–25–11–2–46). We immediately surmised that inappropriate episode references were likely, due to the difficulty of checking whether the deeply-nested episodes fit in the context in which they appear several levels up. Examination showed that many were questionable. Two are evident from the diagram: episode 2 “Retrieve Existing Quote” refers to the undefined episode “Choose Quote” (*d* in the diagram);

Use Case # 11: Edit Quote
Overview:
A Salesperson wishes to edit information contained in a Quote.
Preconditions:
1. Salesperson has successfully logged into Euronet system
2. Euronet Main Screen is displayed on PC

Main Scenario:
1. Salesperson selects Quotes function: Euronet displays Main Quote Screen
2. Salesperson finds quote to edit
   A. Use Retrieve Existing Quote
3. If the Salesperson needs to edit customer information, the Salesperson requests to edit the Quote Header.
   A. Use Edit Quote Header
4. If the Salesperson needs to edit line item information, the Salesperson requests to edit a Line Item
   A. Use Edit Existing Item
5. Salesperson indicates that the quote is complete after all edits are finished
   A. Use Close Quote

Scenario Notes: *[none]*
Post Conditions:
Euronet DB contains an updated quote containing the changes that the Salesperson entered.
Required GUI:
A. Euronet Main Screen
B. Main Quote Screen

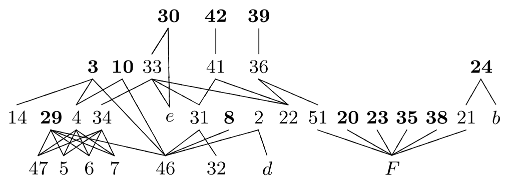
Secondary Scenarios: *[none]*
Uses/Extends: *[none]*
Other Requirements: *[none]*
Revision History:
1. Initial Document Release
   A. Date: 11-March-1999
   B. Description: Initial release
   C. Name: *[author]*

Fig. 16. Euronet use case 11 “Edit Quote”.

and episode 25 “Approve Quote” includes episode 2 twice, once directly and a second time indirectly through 11 “Edit Quote” (Fig. 16). Careful examination shows that the effect is to retrieve an existing quote, and then immediately retrieve the same quote again before editing its header (using the undefined episode “Edit Quote Header”, *i* in the diagram). We believe deep paths (longer than 3) indicate likely inconsistencies between the episodes being used and the context they are used in.

The episode reference diagram also has two clusters that draw the eye, and both turned out to be problematic upon closer examination. For example, the cluster on the left has several episodes or scenarios all referring to the same four episodes: 47 “Get Ship-To Address”, 5 “Get Header General Information”, 6 “Get Header Terms”, and 7 “Get Header Notes”. These four episodes are only used together, and always in the same sequence. Totalling 12 events, it is unclear why they are separate episodes rather than being combined into a single episode that would be easier to understand, keep coherent, and work with. Since we had no access to Euronet stakeholders or analysts it was not possible to resolve this question, and we cannot rule out the existence of some rationale for having four separate episodes. However, it seems likely that they should be combined into one.

Indirect confirmation that deep paths and clusters can indicate problems comes from the episode reference diagram after the application of our approach, which appears in Fig. 17. Use of our approach evolved the scenarios so that the diagram became visually simpler, with one cluster and most deep paths gone (compare Fig. 6).



30 **42** **39**
3 **10** 33 41 36 **24**
14 **29** 4 34 *e* 31 **8** 2 22 51 **20** **23** **35** **38** 21 *b*
47 5 6 7 46 32 *d* *F*

Fig. 17. Euronet episode reference diagram after our approach.

Constructing an episode reference diagram manually is so tedious and time-consuming as to be impractical as a common practice, so automation based on our scenario structure supports an analysis that shows evidence of improving scenario quality but is impractical to do by hand.

5.3.2.6. Automated episode identification finds duplication that people have not. Although SMaRT’s similarity measure and episode identification support was not implemented at the time we performed this study, we manually identified a number of similar scenarios and possible episodes. For example, *UC* “Change Language” partially duplicated a secondary scenario of *UC* “Log On”. We uncovered a number of similar duplications while manually performing episode identification, indicating that episode identification and shared event analysis can improve the minimality of a scenario collection. The manual process was too tedious and lengthy to be practical as an ordinary requirements practice, so that its automation makes possible analyses that improve quality but are not practical without automated support.

5.3.2.7. Our approach is more effective and efficient than GBRAM goal-based analysis. The scenarios our approach yielded were more coherent and provided better coverage than the initial ABB use cases and also than the scenarios produced by the earlier goal-based analysis. Both our approach and GBRAM produced comparable results in terms of identifying simple errors, inconsistencies, and missing information in the initial use cases. However, the gaps in coverage identified by our approach were more substantial and of greater significance than those identified by the goal-based analysis. For example, scenario context diagrams helped us uncover new scenario S98 "Log Off" that defines the events necessary for a user to log off from Euronet, the contexts in which logging off can occur, and the context produced by a user logging off. This was typical for the weight and scope of new scenarios identified using our approach. On the other hand, GBRAM goal-based analysis of the goal <`ENSURE adjustments for pricing for a standard product provided`> yielded seven secondary scenarios to represent possible pricing adjustment alternatives:

- Adjustment determined by competitive level
- Adjustment determined by item quality
- Adjustment determined by rush delivery
- Adjustment determined by sales channel
- Adjustment determined by special features
- Adjustment determined by plant loading
- Adjustment determined by end customer

These were typical of the weight and scope of new scenarios identified using GBRAM. The scope and significance of these scenarios is considerably smaller than that of the scenarios uncovered using scenario context.

In addition, applying our approach to the specification is substantially more efficient than applying GBRAM, required approximately half the analyst-hours that were needed for the goal-based analysis. With more completely implemented tool support, we expect our approach would be even more efficient by comparison.

5.3.3. Discussion

Our experience in the Euronet study confirmed again that attribute and term glossaries and cross-references support well-definedness.

We found that the effectiveness of glossaries for supporting well-definedness, minimality, and coherence was increased by standardizing on a relatively small number of standard words and phrases. The smaller number of terms made it simpler to find the term that was needed, by reducing the number of nearly-synonymous terms that need to be searched among. Careful and insightful choice of the standard terms can help the scenarios make the appropriate distinctions and reduce the cognitive load on people reading the scenarios.

The Euronet use cases and scenarios make extensive use of episodes, and this provided a large scope of action for episode reference diagrams. We found that the diagrams supported coherence in two ways: by visually directing analysts' attention to deep paths and clusters that we believe often indicate likely areas of incoherence, and by conveniently showing an analyst all the locations where a particular episode (or group of episodes) is used so they can be checked for consistency.

Identifying events shared among scenarios was shown to be effective in minimizing the number of distinct events, episodes, and scenarios, encouraging their reuse and supporting minimality of the scenarios.

Finally, we confirmed again that scenario context diagrams are effective in directing attention to gaps in coverage and incoherence, as was already supported in the EMS study.

6. Related work

A number of researchers have noted the need for automated support in working with scenarios. Maiden describes the CREWS-SAVRE tool for semi-automatic generation of scenarios from the actions of use cases [46]. Breitman et al., present SET (Scenario Evolution Tool) which supports the orderly evolution of collections of scenarios, based on a model of fundamental relations and operations on scenarios. Letier et al., use the LTSA tool to identify possibly undesired system behaviors, in the form of implied scenarios, that are difficult to find manually [45]. Woo and Robinson use tool support to identify similar scenarios [64]. Our approaches are distinct from those of these researchers. The goals and capabilities of our SMaRT tool are different from those of the listed scenario tools; we have found no other tools with which we can directly compare it.

Similarity has been used extensively for requirements reuse; for example, semantic similarity has been used with conceptual graphs [57,64], and analogical reasoning on a pre-existing case base [49] and on generic domain models [48]. Semantic similarity is appropriate for these applications because they reapply the semantic structure of pre-existing requirements to new contexts. Several researchers have worked in the area of syntactic similarity measures for requirements analysis. Natt och Dag et al., examine the syntactic representation of requirements in the form of prose, and use a statistical analysis of the preprocessed text to determine similarity of requirements [50]. Like us, they also use their similarity measure as an indicator of less accessible features, in their case relationships between requirements. Park et al., use similarity measures that examine syntactic similarity between prose requirements (in Korean), again using statistical analysis of preprocessed text [51]. Al-Otaiby et al. independently take a syntactic approach similar to ours; they analyze sentences in the scenarios to extract their grammatical objects, and calculate a somewhat simpler similarity using only those objects [1]. For our approach, we believe syntactic similarity is more appropriate. Our approach reuses entire scenarios or attributes, rather than adapting them to new contexts; syntactic comparisons are easier to understand, we believe, and eliminate the need to build a semantic structure.

Requirements continually evolve, increasing the need to track open issues. A goal-based analysis study highlighted the evolutionary nature of goals, scenarios, and requirements [15]. The scenario evolution framework of Breitman et al. suggests two types of changes that affect scenarios during development: "inter-scenario" evolution, where changes involve a set of scenarios, and "intra-scenario" evolution, where changes concern a single scenario [28,29]. We present a viable strategy to manage the relationships in both inter- and intra-scenario evolution. Our strategy is distinct from that of Breitman and Leite; our strategy extends scenario relationships to include a more general view of context and similarity and a stronger focus on dependencies shown in terms of reuse of episodes and terms.

The issue of scenario management has received attention but has not been addressed effectively. In their survey of the use of scenarios in industry, Weidenhaupt et al., note that the creation, documentation, and validation of scenarios is a substantial effort in itself [63]. Jarke et al., specifically note that little research addresses the problems that arise in managing a large set of scenarios [41]. The Dagstuhl Workshop on Scenario Management [40], the resulting special issue of the *IEEE Transactions on Software Engineering* devoted to scenario management [61], and the related special issue of the *Requirements Engineering Journal* devoted to interdisciplinary uses of scenarios [54] took a broad view of scenario management reflecting the use of the term in other areas such as business forecasting, for example the use of scenarios to manage the unfolding of events or to examine the results of alternate courses of action. However, since that time the use of the term "scenario management" in requirements engineering has shifted to more specifically describe the management of collections of scenarios. Alspaugh et al., defined the term with that meaning and proposed an approach to scenario management based on glossaries, episodes (scenario fragments that appear in several scenarios), and measures of similarity between scenarios [6]. Some of that work is an early version of the work presented here. The approach uses a purely syntactic view of scenarios to support analysts as they work to make their scenarios consistent; trace and maintain dependencies among scenarios; look for scenarios that address a particular behavior; and determine completeness of a group of scenarios [6].

Breitman and Leite define and use relationships between scenarios (overlap, equivalence, and subset) to classify and guide the evolution of scenarios [28]. Their definition of scenario equivalence is "when two scenarios share the same episodes, involve the same actors, but handle different restrictions and exceptions". Alspaugh et al., discuss the importance of dependency relationships between scenarios, and their preservation as the scenarios evolve [6].

Glossaries are needed wherever specialized terms are used, or wherever terms are used with meanings more specific than in their ordinary usage. Their use in technical writing is widespread, indeed almost universal. In the area of requirements engineering, glossaries are widely used and their benefits are generally recognized. We cite a few researchers here in illustration. Heitmeyer et al., discuss the use of glossaries (among other techniques) in the consistency checking of requirements specifications [38]. Weidenhaupt et al., state that glossaries add a common understanding of terms used in scenarios [63].

7. Lessons learned and future work

In this paper, we have presented an approach for supporting more effective work with scenarios. This approach is based on the fact that there are aspects of scenarios that have an implicit structure and meaning and these can be the basis for automated tool support. Using tools to do tedious, exacting, or uninteresting tasks helps eliminate human errors by releasing people to concentrate on more interesting work for which human intelligence is essential and most valuable. The result is scenarios that more effectively explore the space of possibilities, more accurately convey the stakeholders' and analysts' understanding, needs, and tradeoffs, and more effectively highlight the requirements issues that can and should be resolved in terms of requirements rather than being left for developers to sort out in subsequent phases. The resulting improvement in well-definedness, coverage, minimality, and coherence of scenarios are crucial because they increase the likelihood that development projects using them will finish in a reasonable time (or at all) and produce software that meets its stakeholders' needs.

We have validated our approach by persuasion (to use Shaw's terminology [58]), presenting arguments in support of the improvements in scenario quality that can be expected from using our approach; by implementing part of the approach in the SMaRT tool; and by evaluating our approach in the context of two studies on scenarios for two different kinds of systems, in two distinct contexts and with scenarios of two different levels of quality. The validation has provided support for our claims that our approach can produce equivalent or better results than ordinary manual practice, and (with automated support) can do so faster and with fewer errors; that it improves scenario quality by providing useful analyses impractical to do by hand; and that is supports greater expressiveness that can be used to improve coverage and well-definedness. The studies show that our approach produces improvements both in good, thoroughly-reviewed scenarios and in scenarios whose quality is not sufficient to support effective development. In short, our approach is effective in supporting analysts as they bring initially inadequate scenarios up to the level of quality essential for developing satisfactory software in a reasonable time, and as they analyze and improve scenarios that are already of good quality.

Our future work includes implementing tool support for the parts of our approach not supported by SMaRT, and repackaging its components for incorporation into several software tools other than a single standalone scenario tool. We are extending our model of scenario structure and meaning so that it supports some logical extensions of standard scenarios and additional uses of scenarios for tasks not now common. One such extension is to incorporate temporal relations besides sequence and concurrency into scenarios. We are examining the use of such formalisms as Allen's interval algebra [3] and partial orderings of events [10,12]. Another is to combine individual scenarios in order to produce a stronger or more expressive composite specification. Some additional uses of scenarios are in support of more effective testing, and as a basis for interactions in computed social worlds. We are looking into using scenarios as the basis for tracing tests to and from stakeholder requirements, and for evaluating test results more efficiently by incorporating knowledge about goals and plans into testing. We plan to extend the evaluation of our work by using stakeholders and analysts other than the authors to evaluate the results, and by comparing the effectiveness and efficiency of using SMaRT to the effectiveness and efficiency of a comparable software tool.

Acknowledgements

The authors thank Aldo Dagnino of Asea Brown Boveri, BellSouth Telecommunications, and the anonymous referees of a previous submission of this paper. This work was supported by NSF CAREER Award #9983926 and BellSouth Telecommunications CACC Grant #5-30092.

References

[1] T.N. Al-Otaiby, W.P. Bond, and M. AlSherif, Sotware modularization using requirements attributes, in: ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference, 2004, pp. 104–109.
[2] I. Alexander, Introduction: scenarios in system development, in: I.F. Alexander, N. Maiden (Eds.), Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, Wiley, 2004, pp. 3–24.
[3] J.F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (11) (1983) 832–843.
[4] T.A. Alspaugh and A.I. Antón, Scenario networks: A case study of the enhanced messaging system, in: REFSQ'01: Requirements Engineering: Foundation for Software Quality, 2001, pp. 113–124.
[5] T.A. Alspaugh and A.I. Antón, Contrasting use case, goal, and scenario analysis of the Euronet system, in: RE'03: 11th International Conference on Requirements Engineering, 2003.
[6] T.A. Alspaugh, A.I. Antón, T. Barnes, and B.W. Mott, An integrated scenario management strategy, in: RE'99: Fourth International Symposium on Requirements Engineering, 1999, pp. 142–149.
[7] T.A. Alspaugh, A.I. Antón, and L.J. Davis. An empirical study of scenario similarity measures, ISR Technical Report UCI-ISR-03-07, Institute for Software Research, University of California, Irvine, 2003.
[8] T.A. Alspaugh, D.J. Richardson, T.A. Standish, and H. Ziv, Scenario-driven specification-based testing against goals and requirements, in: REFSQ'05: Requirements Engineering: Foundation for Software Quality, 2005, pp. 187–202.
[9] Thomas A. Alspaugh, Scenario networks and formalization for scenario management, PhD thesis, North Carolina State University, Raleigh, NC, September 2002.
[10] Thomas A. Alspaugh, Relationships between scenarios, Technical Report UCI-ISR-06-7, Institute for Software Research, University of California, Irvine, May 2006. Revised Dec. 2006.
[11] Thomas A. Alspaugh, Annie I. Anton, and Abdi Modarressi, Enhanced Messaging System scenarios, July 2001. http://www.ics.uci.edu/~alspaugh/EMSscos-public.html.
[12] Thomas A. Alspaugh, Susan Elliott Sim, Kristina Winbladh, Mamadou Di-allo, Hadar Ziv, and Debra J. Richardson, The importance of clarity in usable requirements specification formats, Technical Report UCI-ISR-06-14, Institute for Software Research, University of California, Irvine, September 2006.
[13] A.I. Antón, Ryan A. Carter, Aldo Dagnino, John H. Dempster, Devon F. Siege, Deriving goals from a use-case based requirements specification, Requirements Engineering Journal 6 (1) (2001) 63–73.
[14] A.I. Antón, C. Potts, A representational framework for scenarios of system use, Requirements Engineering Journal 3 (3-4) (1998) 219–241.
[15] A.I. Antón and C. Potts, The use of goals to surface requirements for evolving systems, in: ICSE '98: 20th International Conference on Software Engineering, 1998, pp. 157–166.
[16] Annie I. Antón, Goal Identification and Refinement in the Specification of Software-Based Information Systems, PhD thesis, Georgia Institute of Technology, Atlanta, GA, June 1997.
[17] Annie I. Antón, Michael McCracken, and Colin Potts, Goal decomposition and scenario analysis in business process reengineering, in: Proceedings of the 6th International Conference on Advanced Information Systems Engineering (CAiSE'94), 1994, pp. 94–104.
[18] Asea Brown Boveri – Electric Systems Technology Institute. Software requirements specification for Euronet v.2, 1999.
[19] F. Basanieri, A. Bertolino, G. Lombardi, G. Nucera, E. Marchetti, A. Ribolini, Cow Suite: a UML-based tool for test-suite planning and derivation, ERCIM News 58 (2004) 30–32.
[20] Camille Ben Achour, Colette Rolland, Carine Souveyet, and Neil A. Maiden, Guiding use case authoring: Results of an empirical study, in: Fourth IEEE International Symposium on Requirements Engineering (RE'99), 1999, pp. 36–43.
[21] K. Benner, M.S. Feather, W.L. Johnson, and L. Zorman. Utilizing scenarios in the software development process, in: IFIP Working Group 8.1 Working Conference on Information Systems Development Processes, 1992.
[22] E. Bertino, G. Guerrini, I. Merlo, and M. Mesiti, An approach to classify semi-structured objects, in: ECOOP'99: European Conference on Object-Oriented Programming, 1999, pp. 416–440.
[23] L.J. Bode, A scenario management case study: measuring scenario similarity in the EMS, Technical Report TR-2002-10, North Carolina State University, 2002.
[24] B. Boehm, Software engineering, IEEE Transactions on Computers 25 (12) (1976) 1126–1241.
[25] B. Boehm, V.R. Basili, Software defect reduction top 10 list, IEEE Software 34 (1) (2001) 135–137.
[26] B.W. Boehm, Software Engineering Economics, Prentice-Hall, 1981.
[27] Barry W. Boehm, Verifying and validating software requirements and design specifications, IEEE Software 1 (1) (1984) 75–88.
[28] K. Breitman and J. Leite, A framework for scenario evolution, in: ICRE'98: Third International Conference on Requirements Engineering, 1998, pp. 214–223.
[29] K. Breitman, J. Leite, D. Berry, Supporting scenario evolution, Requirements Engineering Journal 10 (2) (2005).
[30] F.P. Brooks Jr., No silver bullet: essence and accidents of software engineering, IEEE Computer 20 (4) (1987) 10–19.
[31] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2000.
[32] D.W. Cordes, D.L. Carver, Evaluation method for user requirements documents, Information and Software Technology 31 (4) (1989) 181–188.
[33] D. Damian, J. Chisan, L. Vaidyanathasamy, Y. Pal, Requirements engineering and downstream software development: findings from a case study, Empirical Software Engineering 10 (3) (2005) 255–283.

[34] A.M. Davis, Software Requirements: Analysis and Specification, Prentice-Hall, 1990.
[35] A.M. Davis, A.S. Zweig, The missing piece of software development, Journal of Systems and Software 53 (3) (2000) 205–206.
[36] P.A. Gough, F.T. Fodemski, S.A. Higgins, and S.J. Ray. Scenarios – an industrial case study and hypermedia enhancements, in: RE'95: Second International Symposium on Requirements Engineering, 1995, pp. 10–17.
[37] P. Haumer, Use case-based software development, in: Ian F. Alexander, Neil Maiden (Eds.), Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, John Wiley and Sons, Ltd., 2004, pp. 237–264.
[38] C. Heitmeyer, B. Labaw, and D. Kiskis. Consistency checking of SCR-style requirements specifications, in: RE'95: Second International Symposium on Requirements Engineering, 1995, pp. 56–63.
[39] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard, Object-Oriented Software Engineering: A Use Case Driven Approach, ACM Press, 1992.
[40] M. Jarke, X.T. Bui, and J.M. Carroll, Dagstuhl workshop on scenario management, Dagstuhl Seminar Report 199, Schloss Dagstuhl International Conference and Research Center for Computer Science, February 1998.
[41] M. Jarke, X.T. Bui, J.M. Carroll, Scenario management: an interdisciplinary approach, Requirements Engineering Journal 3 (3-4) (1998) 155–173.
[42] J.B. Kruskal, An overview of sequence comparison, in: D. Sankoff, J.B. Kruskal (Eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, 1983, pp. 1–44.
[43] A. van Lamsweerde, L. Willemet, Inferring declarative requirements specifications from operational scenarios, IEEE Transactions on Software Engineering 24 (12) (1998) 1089–1114.
[44] D. Lauzon and T. Rose, Task-oriented and similarity-based retrieval, in: Knowledge-Based Software Engineering Conference, 1994, pp. 98–107.
[45] E. Letier, J. Kramer, J. Magee, and S. Uchitel, Monitoring and control in scenario-based requirements analysis, in: ICSE '05: 27th International Conference on Software Engineering, 2005, pp. 382–391.
[46] N.A.M. Maiden, CREWS-SAVRE: scenarios for acquiring and validating requirements, Automated Software Engineering 5 (4) (1998) 419–446.
[47] N.A.M. Maiden, S. Minocha, K. Manning, and M. Ryan, CREWS-SAVRE: Systematic scenario generation and use, in: ICRE'98: International Conference on Requirements Engineering, 1998, pp. 148–155.
[48] N.A.M. Maiden, A.G. Sutcliffe, Analogical retrieval in reuse-oriented requirements engineering, Software Engineering Journal 11 (5) (1996) 281–292.
[49] P. Massonet and A. van Lamsweerde, Analogical reuse of requirements frameworks, in: RE'97: Third International Symposium on Requirements Engineering, 1997, pp. 26–39.
[50] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karls-son, Evaluating automated support for requirements similarity analysis in market-driven development, in: REFSQ'01: Requirements Engineering: Foundation for Software Quality, 2001.
[51] S. Park, H. Kim, Y. Ko, J. Seo, Implementation of an efficient requirements-analysis supporting system using similarity measure techniques, Information and Software Technology 42 (6) (2000) 429–438.
[52] C. Potts, Software engineering research revisited, IEEE Software 10 (5) (1993) 19–28.
[53] C. Potts, K. Takahashi, Annie I. Antón, Inquiry-based requirements analysis, IEEE Software 11 (2) (1994) 21–32.
[54] Requirements engineering journal, 1998. Special Issue: Interdisciplinary Uses of Scenarios.
[55] S. Robertson, Scenarios in requirements discovery, in: I.F. Alexander, N. Maiden (Eds.), Scenarios, Stories, Use Cases: Through the Systems Development Life-Cycle, John Wiley and Sons, Ltd., 2004, pp. 39–59.
[56] C. Rolland, C. Souveyet, C. Ben Achour, Guiding goal modeling using scenarios, IEEE Transactions on Software Engineering 24 (12) (1998) 1055–1071.
[57] K. Ryan and B. Mathews, Matching conceptual graphs as an aid to requirements re-use, in: RE'93: International Symposium on Requirements Engineering, IEEE, 1993, pp. 112–120.
[58] M. Shaw. The coming-of-age of software architecture research, in: ICSE '01: 23rd International Conference on Software Engineering, 2001, pp. 657–664a.
[59] The Standish Group. The CHAOS report, 1994.
[60] A. Sutcliffe, Scenario-based requirements engineering, in: RE'03: 11th International Conference on Requirements Engineering, 2003, pp. 320–329.
[61] Ieee transactions on software engineering, December 1998. Special Issue: Scenario Management.
[62] A. Tversky, Features of similarity, Psychological Review 84 (4) (1977) 327–352.
[63] K. Weidenhaupt, K. Pohl, M. Jarke, P. Haumer, Scenarios in system development: current practice, IEEE Software 15 (2) (1998) 34–45.
[64] H.G. Woo and W.N. Robinson, Reuse of scenario specifications using an automated relational learner: a lightweight approach, in: RE'02: International Conference on Requirements Engineering, 2002, pp. 173–180.
[65] Didar Zowghi and Vincenzo Gervasi, The 3 Cs of requirements: Consistency, completeness, and correctness, in: 8th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '02), September 2002.