Using Scenario Networks for Scenario Management and Software Specification

TR-2001-11

Thomas A. Alspaugh and Annie I. Antón
College of Engineering
North Carolina State University
Raleigh, NC 27695-7534 U.S.A
{taalspau,aianton}@eos.ncsu.edu

December 4, 2001

Abstract

Scenarios are widely used to specify desired system behaviors, but analyzing and managing large collections of scenarios remain a challenge. Scenario networks facilitate scenario management and serve as a powerful basis for analyzing and validating collections of scenarios. In a scenario network, each scenario is connected to those that may follow it, either as part of a single sequence or concurrently with others. A scenario network provides an integrated specification of an entire system and incorporates behaviors that span two or more scenarios. We show how the process of creating a scenario network improves the quality of the component scenarios by helping analysts identify gaps and inconsistencies that more traditional reviews and walkthroughs normally do not uncover.

Keywords

Scenario management, scenario networks, requirements engineering.

1 Introduction

Scenarios have proven useful in requirements engineering for identifying, elaborating, validating, and refining requirements [4, 8, 9, 10, 11, 13, 16, 17]. Managing large sets of scenarios is challenging, as observed in a study of scenario usage in European industrial projects [18]. In particular, this study revealed that most developers lack appropriate process guidance. In this paper we discuss the successful use of scenario networks for facilitating scenario management and ensuring increased requirements coverage by providing directed process guidance to developers. A scenario network is a set of scenarios and the connections from each scenario to those that may follow it, either in a single sequence or concurrently; any allowable behavior of the system corresponds to a (possibly ramified) path through the network, beginning at an initial scenario and continuing until a terminal scenario is reached. A scenario network thus provides a specification of an entire system and an excellent organizing structure for a collection of scenarios. The scenario relationships afforded by scenario networks allow scenarios to be grouped together in equivalence classes, yielding two ways of organizing scenarios: according to each scenario's place in the overall network, and according to classes of scenarios that are equivalent in some way. The process of deriving and constructing scenario networks provides process guidance for developers since the scenario network serves as the basis for a systematic walkthrough of the expected, desired, and allowable system behaviors. Construction of a scenario network compels analysts to consider alternatives that typically are overlooked, and to find any disallowed alternatives that can be verified as incorrect during walkthroughs.

This work has been validated within the context of a real-world application, BellSouth Telecommunications's Enhanced Messaging System (EMS). The EMS is a comprehensive voice messaging system that supports a wide range of functionality including: access and authentication; configuration management; 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); as well as recording, playing, and archiving of subscriber outgoing messages. An initial version of the EMS scenarios and requirements appear in [2]. This paper is based upon the completed set of scenarios and corresponding requirements.

Our previous work in syntactic relationships employed syntactic relationships to maintain consistency, express interdependence, and identify duplicate, partially-elaborated, and missing scenarios [1]. Our more recent work extends that line of research via the use of semantic relationships among scenarios [2]. In this paper, we demonstrate that scenario networks provide an effective means to formulate a consistent and correct set of scenarios, for subsequent operationalization into requirements.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of relevant work. Section 3 introduces scenario networks, employing an example from the EMS. Section 4 describes the process of constructing a scenario network. In Section 5 we discuss the benefits of applying scenario networks to support scenario management and directed walkthroughs. Finally, the lessons learned and our plans for future work are presented in Sections 6 and 7, respectively.

2 Related work

In this section we briefly discuss the relevant related work on scenario management and process guidance as well as scenario sequences and concurrency.

Scenario management and process guidance

As previously mentioned, scenario management remains a significant challenge to developers [18]. Most project developers lack guidance for deciding the appropriate level of abstraction as well as when to stop. In fact, most developers viewed scenario creation as a craft rather than an engineering task. With scenario networks, scenario creation becomes more systematic and less of a craft. Current approaches to scenario management provide frameworks within which to discuss scenarios [18]. The CREWS framework [18] classifies scenarios according to four facets: Purpose, Lifecycle, Contents and Form. The integrated scenario management strategy of Alspaugh *et al.* [1] provides a concrete strategy for managing large and at times unwieldy amounts of information associated with scenarios. It allows analysts to vary the level of redundancy and consistency checking required for scenario evolution, using similarity measures. The similarity measures provide constructive process guidance to developers, such as stopping criteria for scenario construction.

The benefits of scenario walkthroughs have been previously addressed [12, 14]. Inconsistencies, gaps, and errors in a collection of scenarios for a given system must be resolved at some level before the system's implementation can be satisfactory. Maiden *et al.* attack the problem of missing scenarios with a method supported by their CREWS-SAVRE tool. In this method, new scenarios are automatically generated for consideration by an analyst, using a library of standard models and alternative sequences of use case events [12].

A number of other researchers have attached pre- and postconditions (or initial and final states) to scenarios. Rolland and Ben Achour use initial states of agents and final states of episodes to guide the writing of use cases [15]. Rolland *et al.* attach initial and final states (analogous to pre- and postconditions) to scenarios in their *L'Ecritoire* tool in support of a heuristic to guide the search for additional goals [16].

Scenario sequences and concurrency

A number of researchers have examined the issue of sequence and concurrency among scenarios, and the integration of scenarios into a larger entity that expresses sequence and concurrency.

A use case map (UCM) integrates scenarios to provide a whole-system specification which expresses sequence and concurrency among scenarios [5, 6]. The scenarios are expressed as causal sequences of responsibilities, and denoted graphically as a graph with responsibilities attached. UCMs are primarily seen as a notation for binding responsibilities to system components, and a bridge between system specification and system design. Visual inspection of UCMs is used to find feature interactions. More recently, UCMs have been used to express whole-system behavior, and formalized by (manual) translation into the specification language LOTOS [3]. The goal of this approach is to detect feature interactions that are caused by unexpected interactions between scenarios.

Dano *et al.* integrate scenarios as part of their work on eliciting and validating requirements. Scenarios are integrated by means of a graph whose arcs express the temporal relationships between the nodes representing the scenarios. The integrated scenarios are used for comparisons against less-formal information from domain experts. The goal of the technique is to elicit new requirements and validate the requirements already found [7].

In the remainder of this paper, we present scenario networks as an alternative (and we believe more effective) approach that focuses on formulating a consistent and correct set of scenarios, and integrating the scenarios into a single specification of system behavior. This work is based on our previous work in syntactic relationships among scenarios, in which we used them to maintain consistency, express interdependence, and identify duplicate, partially-elaborated, and missing scenarios [1]. We extend that line of research here by using semantic relationships among scenarios to improve the quality of scenarios and produce integrated specifications expressing behaviors that may span more than one scenario.

3 Scenario networks

Individual scenarios are frequently used to describe a single transaction or a single sequence of events accomplishing a particular purpose (as in Scenario (S12) shown in Table 1). A scenario describes part of a system's behavior, and a group of scenarios describes the entire behavior of a system. Ideally, every system behavior is expressed by some scenario in the group.

$\mathbf{S_{12}}$. Subscriber listens to the next message.
1. Subscriber S dials the *next new message* command.
2. EMS plays S's next message.
3. If that message was "new", EMS changes its state to "old".

Table 1: EMS Scenario S_{12}

What is not expressed by a group of scenarios is the allowed temporal relationships among all the scenarios. There is no specification of either the allowable sequences of scenarios or concurrency between sequences of scenarios. Scenario networks provide a way to express this additional information.

A *scenario network* is comprised of a group of scenarios and the interconnections between them that indicate the allowed scenario sequences and concurrency. Any allowable behavior of the system corresponds to a (possibly ramified) path through the network, beginning at an initial scenario and continuing until each branch of the path reaches a terminal scenario.

A detailed example of a sequential scenario network for a simplified EMS is presented in our earlier work [2]. Here we present an example from the complete EMS that demonstrates concurrency, using the scenarios listed in Table 2.

In the context of the EMS, expected or desired behaviors are represented by a number of multipaths through these scenarios. A *multipath* is a possibly

S_0	EMS startup.
S_1	EMS shutdown.
S_2	Subscriber calls EMS and authenticates him/herself.
S_{12}	Subscriber listens to the next message.
S_{13}	Subscriber has no more messages to listen to.
S_{29}	Subscriber disconnects from EMS.
S_{30}	Caller calls a subscriber and leaves a message.
S_{39}	Caller disconnects from EMS.

Table 2: EMS scenarios used in the example

ramified path through a scenario network. Wherever a multipath diverges into two or more paths, it indicates concurrency between the scenario sequences on the parallel paths. Some allowed multipaths for the EMS are listed in Table 3 (compressed into a linear sequence to aid readability).

- $S_0 \to S_1$
- $S_0 \to S_2 \to S_{13} \to S_{29} \to S_1$
- $S_0 \to S_{30} \to S_{39} \to S_1$
- $S_0 \to S_2 \to S_{13} \to S_{29} \to S_{30} \to S_{39} \to S_1$
- $S_0 \to S_{30} \to S_{39} \to S_2 \to S_{12} \to S_{29} \to S_1$
- $S_0 \to S_{30} \to S_{39} \to S_2 \to S_{12} \to S_{13} \to S_{29} \to S_1$
- $S_0 \to S_{30} \to S_{39} \to S_{30} \to S_{39} \to S_2 \to S_{12} \to S_{29} \to S_1$
- $S_0 \to S_{30} \to S_{39} \to S_{30} \to S_{39} \to S_2 \to S_{12} \to S_{12} \to S_{13} \to S_{29} \to S_1$

Table 3: Allowed EMS multipaths

The list of allowed multipaths continues without end, so rather than listing the multipaths we create a scenario network that expresses exactly those multipaths that are allowed.

We express scenario networks in one of three ways: in tabular form; as a diagram; and through pre- and postconditions for each scenario. In this paper, we focus on the tabular and diagrammatic forms. The pre- and postcondition form is discussed in [2]. For the tabular form, we list the network's scenarios, identify those that are initial or terminal, and give each scenario's follow set and concurrency set. An *initial scenario* is one that may begin a scenario multipath, and a *terminal scenario* may end a branch of one. A scenario's *follow set* is the set of scenarios that may follow it in a sequence, and its *concurrency set* is the set of scenarios that may begin a new concurrent sequence after it. Table 4 provides this information for the example scenarios drawn from the completed set of EMS scenarios.

A second way to express a scenario network is by producing a diagram in which scenarios, represented by circles, are connected by arrows indicating

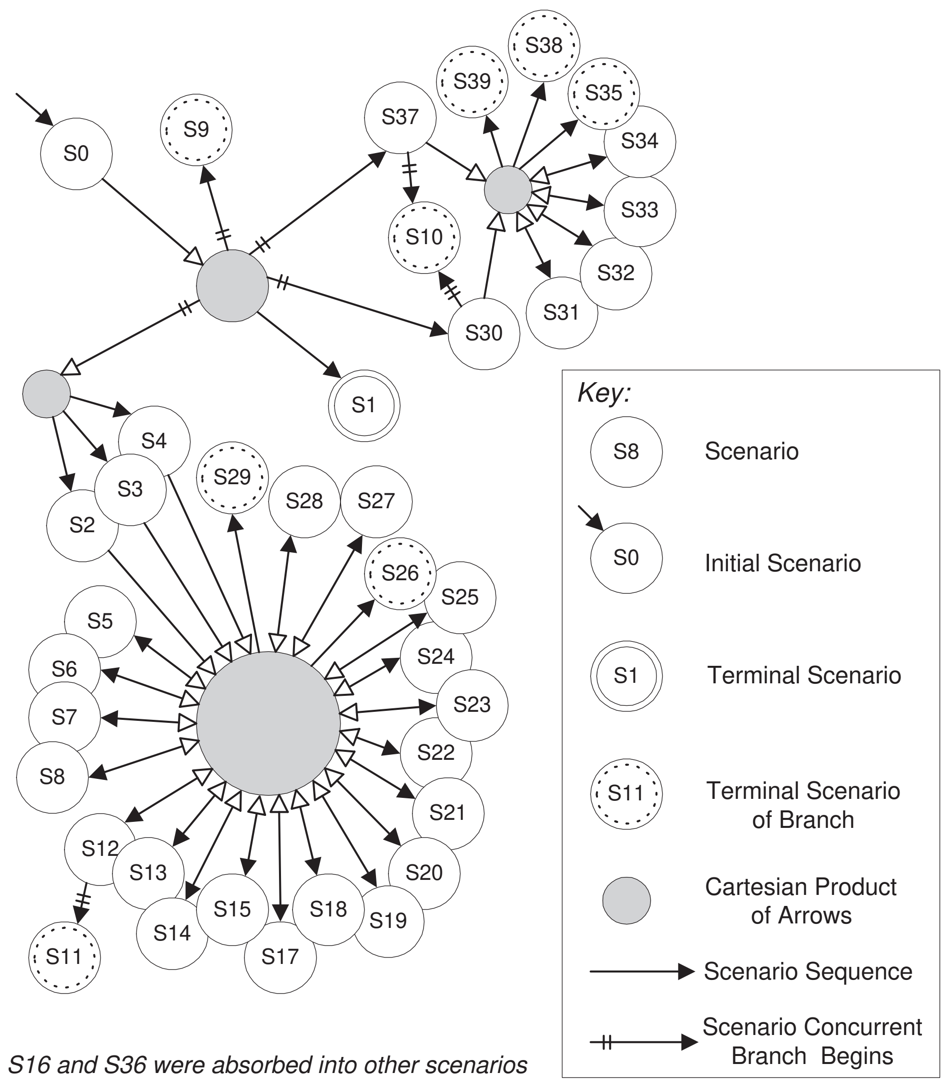

Figure 1: Diagram of EMS scenario network

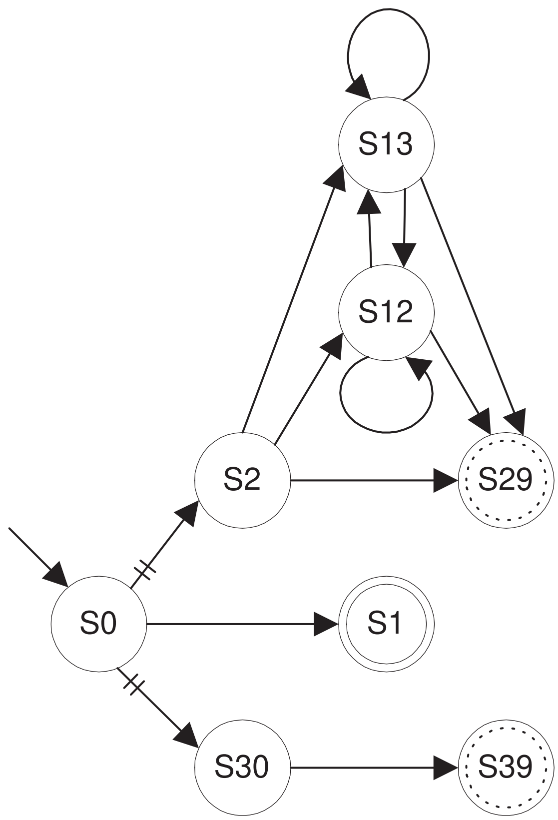

Figure 2: Diagram for example scenario network

Initial scenarios	S_0	
Terminal scenarios	S_1, S_{29}, S_{39}	
Scenario	Follow set	Concurrency set
S_0	S_1	S_2, S_{30}
S_1	$\emptyset$	$\emptyset$
S_2	S_{12}, S_{13}, S_{29}	$\emptyset$
S_{12}	S_{12}, S_{13}, S_{29}	$\emptyset$
S_{13}	S_{12}, S_{13}, S_{29}	$\emptyset$
S_{29}	$\emptyset$	$\emptyset$
S_{30}	S_{39}	$\emptyset$
S_{39}	$\emptyset$	$\emptyset$

Table 4: Tabular form for scenario network

sequence, and slashed arrows indicating where concurrent sequences may begin, as portrayed by the scenario network diagrams in Figures 1 and 2.

Figure 1 shows the entire scenario network diagram for the completed EMS. For purpose of illustration, we focus our discussion on a portion of the scenario network (shown in Figure 2).

The scenario network in Figure 2 supports all the allowed scenario multipaths listed in Table 3, and an infinite number of others. However, it also supports an infinite number of scenario multipaths that should not be allowed, such as

$$S_0 \rightarrow S_{30} \rightarrow S_{39} \rightarrow S_2 \rightarrow S_{12} \rightarrow S_{12} \rightarrow S_{29} \rightarrow S_1$$
(caller left one message but subscriber listened to two)

Such undesired multipaths are ruled out by assigning a precondition to each scenario. A scenario's precondition is required to be true in order for the scenario to begin. Each scenario is also assigned a postcondition which is satisfied at the end of the scenario. Thus at any point in a multipath we can determine which scenarios can occur next by comparing their preconditions with the postcondition fulfilled by the scenario just completed. Our previous work describes the use of these pre- and postconditions, and how they allow the scenario network to serve as an operational model [2].

We note that scenario networks may appear on the surface similar to finite state machines, but there are important differences between them. Notably, each scenario's pre- and postcondition refer to a state that can be arbitrarily complex, so that a scenario network is capable of much more complex transition sequences than a finite state machine. Also, a scenario network addresses concurrency in a manner different from the way a finite state machine does nondeterminism.

4 Constructing Scenario Networks

In this section, we discuss two systematic approaches to construct a scenario network for a collection of scenarios: (1) annotating each scenario with pre- and postconditions, and (2) refining large-scale narratives covering many scenarios about the system's behavior into smaller units.

The starting point of the pre- and postcondition approach is to consider what result each scenario achieves, and what it requires in order to achieve that result. For the EMS, we expressed these results and needs as formal pre- and postconditions for each scenario [2]. These conditions express a sequential network for the scenarios because each scenario's precondition is satisfied by exactly the scenarios that can precede it. A table or a diagram of the scenario network can then be produced, although in order to distinguish concurrency more information is needed. This approach makes automated support possible for determining the relationships between the conditions, and thus the relationships between the scenarios with those conditions. However, without such automated support determining these relationships is cumbersome for the analyst. Additionally, even with a completed set of pre- and post-conditions for each scenario, analysts must still determine concurrency as an additional task because the pre- and post-conditions do not express concurrency. The EMS case study showed that this was not an effective method for constructing a scenario network [2] since simply relying on pre- and post-conditions does not take advantage of the analyst already knows about desired sequencing and concurrency. For example, due to our familiarity with how voice messaging systems operate, we are able to draw upon tacit knowledge. We know that concurrency is possible since a caller can leave a message for a subscriber at precisely the same time at which the subscriber may be listening to other voice mail messages in their mail box. But the pre- and postconditions for these scenarios did not indicate this concurrency. In the remainder of this paper, we discuss the narrative refinement approach, which proved much more effective.

In the narrative refinement approach, analysts consider the envisioned system that is described by the collection of scenarios. The system's behavior is first described with a single high-level story, expressed in natural language. For example, in the EMS case study the single high-level story was "The EMS records messages that callers leave for particular subscribers; subscribers may listen to these messages and store them for a time if they wish." The stories are then successively decomposed into smaller, more detailed stories until each story contains enough detail so that all the desired system behaviors are described in some story. These smaller stories may be extracted from the higher-level stories in several ways:

- choosing a single actor and following his/her interactions with the system;
- following a single concurrent thread of behavior of the system and its environment; or

- identifying alternatives to a previously identified path of action(s).

As the stories become progressively more detailed they yield sequences of the specific scenarios covering actions that take place in that story. The stories and the sequential, alternative, and concurrent relations among their component scenarios form a formidable basis for a scenario network diagram.

Other approaches are possible and constitute future work. For example, these two approaches (annotating with pre- and post-conditions, and refinement of narratives) assume that a collection of scenarios already exists before the scenario network is begun, but in practice this need not be the case. We have seen indications that an approach in which the scenarios and the scenario network are constructed in parallel can offer further advantages. In this situation, the scenario network provides process guidance throughout the creation of the scenarios, rather than only when the scenario collection is nearly complete. Individual scenarios can be produced by elaborating the stories to the desired level of detail, and dividing them into smaller stories and ultimately scenarios. The additional effort required for constructing the scenario network would be minimal with this approach.

5 Applying Scenario Networks

Scenario networks offer many benefits to developers as they seek to specify a system's requirements. A scenario network records the sequences and concurrency that can occur among its scenarios, information that is not part of the individual scenarios and may not otherwise be recorded. A scenario network provides a clear visual representation of those sequences and concurrency, in the form of the scenario network diagram. The process of creating a scenario network helps the analyst improve the quality of the individual scenarios and (our case study showed) also helps the analyst uncover problems with the requirements and thus improve their quality as well (Section 5.2 discusses this in greater detail). A completed scenario network acts as a system specification that includes everything the component scenarios specify and additionally specifies behavior that spans more than one scenario, thus providing a more effective operationalization of the system's requirements. All these benefits aid in the validation of the analysts' understanding of the system's behavior, the system specification (which is the scenario network itself), and the system requirements. In this section, we discuss in more detail how the application of scenario networks during requirements engineering activities offers two specific benefits: scenario networks provide a framework for scenario management and offer process guidance to analysts, as discussed in sections 5.1 and 5.2, respectively.

5.1 Scenario management

Scenario networks provide a rich framework for managing a collection of scenarios. Additionally, the equivalence relations between scenarios that are uncovered during the creation of a scenario network form a useful means for classifying and linking the scenarios in a collection.

A scenario network provides a temporal scheme for organizing scenarios; every scenario has a place in the network. A scenario's place in the scheme is based upon where it can occur, temporally, relative to the other scenarios. Temporal position is an important aspect of the scenario from the analyst's point of view. Thus it produces a better organization than one based on a more arbitrary aspect, such as the order in which the scenarios were created. Organizing scenarios according to the order in which they were identified make fails to provide insights into meaningful relationships, such as sequencing and concurrency. In contrast, scenario networks highlight missing scenarios and duplication between scenarios in a way that analysts find natural and useful and that lends itself to methodical validation via walkthroughs (discussed in Section 5.2).

A disadvantage of organizing scenarios with a scenario network is that this organization does not linearize easily; its natural topology is that of a graph. Thus a scenario network does not translate straightforwardly into a table of contents, for example.

An assortment of relations between scenarios arise during the construction of a scenario network or are indicated by the structure of a scenario network. The most immediately useful of these are equivalence relations, which indicate scenarios that are equivalent in some sense. Examples are the relation that groups together scenarios that can follow all of the same scenarios (illustrated by Table 5); or scenarios that can precede all of the same scenarios; or scenarios that can both follow the same scenarios and precede the same scenarios. Formally, each of these equivalence relations partitions a collection of scenarios into disjoint equivalence classes; the scenarios in each class are interchangeable in the sense of the relation, and every scenario is in exactly one class. Informally, each relation groups scenarios that are similar in a particular way. As an example, Table 5 presents a group of equivalence classes for the "follows the same scenarios" relation. In this example, S_{12}, S_{13}, and S_{29} form an equivalence class because each of them can follow only S_2, S_{12}, or S_{13}. Each of the eight scenarios appears in exactly one equivalence class.

Follow-equivalence classes		
follow $\emptyset$	:	S_0
follow $\{ S_0 \}$	:	S_1, S_2, S_{30}
follow $\{ S_2, S_{12}, S_{13} \}$	:	S_{12}, S_{13}, S_{29}
follow $\{ S_{30} \}$	:	S_{39}

Table 5: Equivalence classes example

These equivalence classes of scenarios are useful in scenario management. Any change to one scenario in a class is likely to be needed for the other scenarios in the class, or at least is more likely to be needed than for a scenario unrelated to the changed one. The scenarios in a class are consistent in some way (or should be if they are not); specifically, scenarios that are precede-equivalent produce the same or similar results, and scenarios that are follow-equivalent require the same or similar things in order to achieve their results. For example, looking at Table 5 we can see that a change in scenario S_{12} may also require corresponding changes in the scenarios S_{13} and S_{29} that are follow-equivalent to it, and that the three scenarios have the same or similar prerequisites (in fact, all three scenarios require that the subscriber in question already be authenticated to the EMS by his or her passcode, with S_{12} additionally requiring that this subscriber have more messages to hear and S_{13} additionally requiring that he or she have no more messages).

5.2 Process guidance

Scenario networks offer process guidance in several ways.

- A scenario network serves as a useful visual aid for guiding walkthroughs of the scenarios. The benefits of walkthroughs are widely known [12, 14].
- Perhaps surprisingly, the process of constructing a scenario network provides important benefits besides the obvious one of producing the network. Constructing a scenario network by either of the systematic approaches described in Section 4 guides the analyst in identifying gaps in coverage; identifying redundancy among the scenarios in the form of overlap and duplication; recognizing equivalence relations between the scenarios and the consistency among scenarios that is required by those relations; and establishing what each scenario requires and accomplishes.
- The graphical properties of a scenario network offer guidance in identifying stopping points for the process of seeking additional scenarios: stop when the scenario network no longer has gaps.

We discuss some of these points in more detail below.

Scenario walkthroughs

A scenario network diagram provides a graphical aid that is well suited to walkthroughs. The visual layout of the diagram suggests paths to follow and provides guidance in the choice of what paths to traverse or walk through next. Overlaps and duplications between scenarios are highlighted during walkthroughs whose course is directed by the visual form of the scenario network. This form can be especially useful in indicating alternatives that may not have been previously considered, because these alternatives are visually obvious in the diagram. Potential concurrency between scenarios, which is frequently difficult to uncover, is also indicated visually by a scenario network diagram. Finally, the diagram provides a form in which certain kinds of information uncovered by the walkthrough can be conveniently recorded as the walkthrough proceeds.

We note particularly that without a scenario network, there may be no obvious document in which to record behavior and relationships that span several scenarios

Guidance from systematic construction

The process of constructing a scenario network requires a continuing series of walkthroughs to check the part of the scenario network completed so far and to examine how remaining scenarios should be added to the network. Our case studies indicate that an analyst naturally tends to check each change to his/her scenario network by walking through several characteristic narratives for multipaths that pass through the scenarios or connections that were changed. Similarly, an effective way to determine how to add a scenario to a network is to walk through some narratives that involve that scenario. For example, consider the process of adding the scenario S_{37} "Caller calls EMS directly and leaves a message" to the scenario network of Figure 2. Walking through the paths that could involve this scenario shows that S_{37} is quite similar (follow- and precede-equivalent) to S_{30} "Caller calls a subscriber and leaves a message." Thus S_{37} should be connected into the diagram in exactly the ways that S_{30} is. When we examine the full EMS diagram in Figure 1, we see that this is indeed the case. We also find that for an analyst, thinking about the system in terms of walkthroughs suggests additional new multipaths that should be walked through. These walkthroughs offer opportunities to improve the quality of individual scenarios and the analyst's understanding of the system as recorded in the scenario network.

Another benefit that occurs as a scenario network is constructed is the uncovering of equivalence classes of scenarios. Construction of a scenario network forces the analyst to consider which scenarios are equivalent and to what degree. This occurs as the analyst considers which scenarios can follow, precede, or begin concurrently with a particular scenario, as part of the process of creating a scenario network diagram. The diagram expresses these relationships graphically, and a correct diagram is one in which (among other things) each of these relationships is expressed correctly. The careful analysis that occurs during the construction of a scenario network and the resulting clarifications of and corrections to the scenarios also improve the quality and usefulness of the scenarios and results in a more complete understanding of how the system is to behave. Our case study indicated that some of the corrections to the scenarios also resulted in corrections to the requirements as well.

The analyst's consideration of scenario equivalence during the construction of a scenario network results in specific attention to what each scenario assumes is available and what each scenario assumes is true, and to the results the actions of each scenario produce. Construction of a scenario network necessarily draws attention to gaps that correspond to missing scenarios, to overlaps between scenarios (of the actions they contain, or the results they produce, or in what they assume), and to unsuspected episodes shared by several scenarios or even to the containment of one scenario by another.

A stopping criterion

Gaps in the scenario network usually correspond to missing or incomplete scenarios. The visual metaphor provided by the diagram makes the discovery of such unsuspected scenarios much easier than it would be otherwise. For example, construction of the EMS scenario network uncovered the missing scenario that became S_{13}, "Subscriber has no more messages to listen to." It was found because, as in Figure 2, S_{12} could be repeated (as indicated by the arrow from S_{12} to itself) but there was no different behavior specified to occur when SSHearMsg could no longer be repeated because there were no more messages to hear.

6 Lessons learned

We learned several lessons during the course of the case study and the work that has followed it.

A scenario networks are a more effective specification than a collection of scenarios

We found that a scenario network is a richer, more expressive, and more effective specification than a collection of scenarios. A scenario network expresses not only the behaviors that each of its scenarios expresses, but also larger-scale behavior spanning several scenarios, which a simple collection of scenarios cannot express. A scenario network constitutes a single unified specification, in which (if the network is properly made) the constituent scenarios are made consistent with each other, gaps in coverage of the system have been eliminated, and the boundary between what the system does and does not do has been delineated. These characteristics of a scenario network remove some of the most significant problems that arise when a collection of scenarios is used as a specification.

Scenario networks validate scenarios and requirements

Our case study showed that scenario networks are effective in validating scenarios and requirements. This validation occurs during the construction of a scenario network, during walkthroughs of a full scenario network, and through consideration of the relations between scenarios that a scenario network diagram expresses graphically, as we discussed at greater length in Section 5.2.

One of the most exciting results of our use of scenario networks in the case study was the discovery of two errors in the EMS requirements through examination of the scenario network. Specifically, this occurred during the validation of the pre- and postconditions for a scenario "Subscriber listens to an archived message" (deleted from the final set of scenarios) that presented a subscriber's archived messages. Tracing this scenario back to the requirements that it operationalized revealed that those requirements were incompatible with other EMS requirements, and incidentally that another requirement, for the sequence of presentation of all messages, did not match what the system's sponsor had requested. Neither of these errors had been discovered during extensive reviews of the requirements.

A scenario network provides a unified viewpoint

Since an individual scenario can have a viewpoint, and the scenarios in a collection will in general not have the same viewpoint, it was not initially clear whether a scenario network could be said to have a viewpoint and if so how that viewpoint could be characterized. We characterize this viewpoint as "omniscient", in keeping with literary usage which contrasts the viewpoint of an individual narrator with a viewpoint that is from no one character's point of view, and that may incorporate any fact or insight that is of interest regardless of which characters (if any) in the story could know them. Similarly, while an individual scenario is commonly expressed from the viewpoint of a particular actor, a network containing that scenario incorporates all actions and events that are needed to describe the system, regardless of whether they are visible from the viewpoint of a single actor.

For a collection of scenarios that are written in terms of actions, events, and effects that are all externally visible, the corresponding scenario network will also be expressed in externally visible terms. In this particularly useful case, the omniscient viewpoint of the scenario network can also be described as the viewpoint that takes in all externally visible phenomena involving the system.

Pre- and postconditions provide benefits

We found that pre- and postconditions for scenarios provided several specific benefits.

A scenario's pre- and postcondition are a very specific representation of the result of that scenario, and what that scenario requires to produce its result. In our case study, creation of these specific representations of a scenario's results and prerequisites focused the attention of the analysts on aspects of the scenarios, and the requirements they traced back to, that until then had escaped careful scrutiny. The precise form of the pre- and postconditions provided guidance to the analysts as they elaborated the precise effect of each scenario.

The conditions also give a precise statement of when each scenario can occur, more restricted than that given by the connections between scenarios in the tabular and diagrammatic forms; these connections are occasionally not stringent enough to express exactly the sequential and concurrent relations between particular scenarios.

The pre- and postconditions incidentally provide an additional form against which to validate the temporal relationships expressed in informally drawn scenario network diagrams, stories linking several scenarios, or specific system requirements.

An informal scenario network is still useful

While pre- and postconditions on scenarios provide definite benefits in the context of a scenario network, they also require a substantial effort and are more formal than many analysts will be comfortable with. We learned that an analyst can relatively quickly create an informal scenario network diagram, without pre- and postconditions for the scenarios. We then observed that even without the additional information provided by pre- and postconditions, informal scenario network diagrams provide substantial results to the analyst. The visual form of the scenario network diagram contains most of the information about a scenario network in a form that is easily used for walkthroughs and expresses relations between scenarios graphically. We found that an informal scenario network diagram was useful in every way that a scenario network diagram is, except for the few ways that directly use the pre- and postconditions.

Scenario networks are a basis for scenario management

As we discuss in detail in Section 5.1, scenario networks provide an organization for a collection of scenarios, and the scenario relations they embody support scenario management as well.

Scenario networks provide process guidance

As we discuss in detail in Section 5.2, scenario networks also provide process guidance in a number of ways. They aid in directing effective walkthroughs of a collection of scenarios; their systematic construction guides discovery of gaps, overlap, and duplication; and a scenario network gives a much-needed stopping criterion for scenario development.

In our case study, scenario networks proved useful in identifying missing scenarios that were needed to complete some system function that was partially given in a scenario already present, or to make the function provided by one group of scenarios in the network parallel to that provided by another. An example of a partially missing function is S_{13} "Subscriber has no more messages to listen to," which was needed to complete the function that S_{12} "Subscriber listens to a message" provided; an example of a missing parallel function was S_{38} "Caller takes no action for a long time," which was added to match S_{26} "Subscriber takes no action for a long time."

7 Future work

We have seen that the viewpoint of a scenario network is of interest. Further work is needed to determine whether a scenario network can be of use in viewpoint issues that affect individual scenarios, such as in recasting a scenario into another viewpoint.

While pre- and postconditions provide unique benefits as part of scenario networks, they proved relatively difficult to work with. Further investigation into how they may be worked with more efficiently, including the possibilities of automated assistance for various specific forms of conditions, is needed.

In our work thus far, we have had success with sequential composition, in which we use pre- and postconditions of scenarios to compose two or more scenarios in a sequence into a larger entity with its own appropriate pre- and postcondition. Concurrent composition, and sequential composition in the presence of arbitrary concurrency, presents much more serious difficulties. We are investigating how to identify interactions between conditions due to concurrency, and what steps can be taken to extend composition into these areas as well.

We have two equivalent forms in which to express a scenario network, one tabular and one diagrammatic. A third form, that of the pre- and postconditions for the scenarios, is only partially equivalent as it does not express concurrency. The scenario networks we have produced thus far have involved relatively simply pre- and postconditions and system states, and for these networks concurrency did not have to be visible in the pre- and postconditions. We plan to examine whether the concurrency should be visible there, and if so how it would best be evident.

In our treatment of concurrency so far, we have concentrated on what is most appropriate for its use in requirements engineering. We plan a more thorough application of the existing theory and body of work on concurrency to scenario networks.

Our research has identified a growing number of scenario relations that arise with a scenario network. We have found practical uses for several of these relations, and expect that further research may find others.

We have located several sets of scenarios from industrial projects and are making plans to use them in further case studies. Another area of interest for these case studies is the comparison of the results achieved by analysts not already familiar with scenario networks, and the simultaneous development of scenarios and scenario networks as discussed in Section 4. We are interested to see to what degree different analysts create similar scenario networks to describe the same system, and we hypothesize that the use of a systematic approach, such as narrative refinement, will tend to result in a convergence among scenario networks describing a particular system.

8 Acknowledgements

The authors wish to thank Abdi Modarressi of BellSouth Telecommunications, Inc. in Atlanta, Georgia for discussions leading to some of this work. This work was partially funded by BellSouth Telecommunications CACC Grant #5-30092, and partially funded by NSF CAREER Grant #530195.

References

[1] T. A. Alspaugh, A. I. Antón, T. Barnes, and B. Mott. An integrated scenario management strategy. In *RE '99: IEEE Fourth International Symposium on Requirements Engineering*, pages 142–149, June 1999.

[2] T. A. Alspaugh and A. I. Antón. Scenario Networks: A case study of the Enhanced Messaging System. In *REFSQ'01: Seventh International Workshop on Requirements Engineering: Foundation for Software Quality*, June 2001.

[3] D. Amyot, L. Logrippo, R. Buhr, and T. Gray. Use Case Maps for the capture and validation of distributed systems requirements. In *RE '99: IEEE Fourth International Symposium on Requirements Engineering*, pages 44–54, June 1999.

[4] A. Antón and C. Potts. The Use of Goals to Surface Requirements for Evolving Systems. In *ICSE'98: IEEE 20th International Conference on Software Engineering*, pages 157–166, April 1998.

[5] R. Buhr and R. Casselman. *Use case maps for object-oriented systems*. Prentice Hall, 1996.

[6] R. Buhr. Use Case Maps as Architectural Entities for Complex Systems. In *IEEE Transactions on Software Engineering*, 24(12), pages 1131–1155, Dec. 1998.

[7] B. Dano, H. Briand and F. Barbier. An approach based on the concept of use case to produce dynamic object-oriented specifications. In *RE '97: IEEE Third International Symposium on Requirements Engineering*, pages 54–64, 1997.

[8] P. Haumer, K. Pohl and K. Weidenhaupt. Requirements Elicitation and Validation with Real World Scenes. *IEEE Transactions on Software Engineering*, pages 1036–1054, Dec. 1998.

[9] H. Kaindl. A Design Process Based on a Model Combining Scenarios with Goals and Functions. *IEEE Transactions on Systems, Man and Cybernetics*, 30(5), pages 537–551 Sep. 2000.

[10] A. van Lamsweerde, R. Darimont and E. Letier. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, pages 908–925, Nov. 1998.

[11] J.C. Leite, G. Rossi, F. Balaguer, V. Maiorana, G. Kaplan, G. Hadad and A. Oliveiros. Enhancing a Requirements Baseline with Scenarios. *Requirements Engineering Journal*, 2(4), pages 184–198, 1997.

[12] N. Maiden, S. Minocha, K. Manning, and M. Ryan. CREWS-SAVRE: Systematic scenario generation and use. In *ICRE '98: Third International Conference on Requirements Engineering*, pages 148–155, 1998.

[13] C. Potts. Using Schematic Scenarios to Understand User Needs. In *Proc. ACM Symposium on Designing Interactive Systems: Processes, Practices and Techniques (DIS'95)*, Aug. 1995.

[14] C. Potts, K. Takahashi and A.I. Antón. Inquiry-Based Requirements Analysis. *IEEE Software*, 11(2), pp. 21-32, Mar. 1994.

[15] C. Rolland and C. Ben Achour. Guiding the construction of textual use case specifications. *Data and Knowledge Engineering Journal*, 25(1–2): pages 125–160, Mar. 1998.

[16] C. Rolland, C. Souveyet, and C. Ben Achour. Guiding goal modeling using scenarios. *IEEE Transactions on Software Engineering*, 24(12): pages 1055–1071, Dec. 1998.

[17] A. Sutcliffe. Scenario-Based Requirements Analysis. *Requirements Engineering Journal*, 3(1): pages 48-65, 1998.

[18] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer. Scenarios in System Development: Current Practice. *IEEE Software*, 15(2): pages 34–45, Mar/Apr. 1998.