Scenario Networks: A Case Study of the Enhanced Messaging System

Thomas A. Alspaugh and Annie I. Antón

North Carolina State University, Department of Computer Science,
Raleigh NC 27695, USA
taalspau@unity.ncsu.edu

Abstract. Scenarios are widely used to specify desired system behavior. In this paper, we discuss a case study of an enhanced voice messaging system, in which the scenarios describing it were assembled into a scenario network. In a scenario network, each scenario is connected to those that may follow it. The resulting scenario network provides a specification of the entire system. The process of creating the scenario network improved the quality of the resulting specification by enabling us to identify gaps and inconsistencies that reviews and walkthroughs had not uncovered. Production of a scenario network compels analysts to improve the coverage and correctness of a set of scenarios, thereby improving the requirements engineering process and the resulting documentation.

1 Introduction

Any inconsistencies, gaps, and errors in the set of scenarios for a given system must be resolved at some level before the system's implementation can be satisfactory. In this paper we describe our process for resolving such inconsistencies, gaps, and errors during requirements analysis. We report our experiences using this process for an Enhanced Messaging System (EMS) for telephone voicemail. The EMS is a comprehensive voice messaging system which 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.

In Section 2 we discuss relevant work in the use of scenarios as specifications; in Section 3 we introduce scenario networks and demonstrate their use with an example from the EMS; in Section 4 we describe the EMS case study; and in Section 5 we discuss the lessons learned and our plans for future work.

2 Related work

In this section we briefly discuss some related work on integrating scenarios, expressing pre- and postconditions that could be used to integrate them, and uncovering missing scenarios.

Dano *et al.* integrate scenarios based on the temporal relationships between them, and construct corresponding Petri nets, as part of their work on formalizing domain expert use cases and producing object type state diagrams from them [4]. However, they do not make use of the integrated scenarios.

A use case map (UCM) integrates scenarios to provide a whole-system specification [3]. The scenarios are expressed as causal sequences of responsibilities and denoted graphically as a graph with responsibilities attached. Concurrency is explicit at all levels. The original emphasis was on binding responsibilities to system components and elicitation of requirements and design information. Feature interaction is examined visually by inspection of the UCM notation. More recently, UCMs have been used to express whole-system behavior, and formalized by (manual) translation into the specification language LOTOS [2].

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 [6]. 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 [7].

Maiden *et al.* attack the problem of missing scenarios with a method supported by their CREWS-SAVRE tool, which automatically generates new scenarios for consideration by an analyst, using a library of standard models and generation of alternative sequences of use case events [5].

In the remainder of this paper, we present scenario networks as an alternative (and we believe more effective) approach focused on formulating a consistent and correct set of scenarios at the requirements level. 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 using semantic relationships among scenarios.

3 Scenario networks

An individual scenario typically describes a single transaction or a single sequence of events accomplishing a particular purpose. Such a scenario conveniently describes part of a system's behavior. However, many scenarios are needed to describe the entire behavior of a system, so that every system behavior is expressed in some scenario or set of scenarios. Since a set of scenarios does not express the sequences in which the scenarios can occur, we connect each scenario to all the scenarios that can immediately follow it, producing a scenario network. A *scenario network* is a set of scenarios and the connections from each scenario to those that may follow it; at least one of the scenarios is distinguished as initial, and at least one as terminal. Then any allowable behavior of the system corresponds to a path through the network, beginning at an initial scenario and continuing until a terminal scenario is reached.

As an example, consider this subset of the scenarios for the EMS:

S_0. System startup.

1. Administrator starts up the system.

S_1. Caller leaves a message for subscriber

1. Caller leaves a message and hangs up.

S_2. Subscriber listens to a new message.

1. Subscriber dials the *next new* command.
2. EMS plays his/her next new message.
3. EMS changes the message's state from "new" to "old".

S_3. Subscriber has no more new messages to listen to.

1. Subscriber dials the *next new* command.
2. EMS says he/she has no more new messages and hangs up.

S_4. System shutdown.

1. Administrator issues the shutdown command.
2. EMS stops accepting new calls.
3. EMS shuts down when all calls are completed.

There are a number of sequences of these scenarios that express desired or expected behaviors for the EMS:

- $S_0 \rightarrow S_4$
- $S_0 \rightarrow S_3 \rightarrow S_4$
- $S_0 \rightarrow S_1 \rightarrow S_4$
- $S_0 \rightarrow S_3 \rightarrow S_1 \rightarrow S_4$
- $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4$
- $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4$
- $S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_4$
- $S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_2 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4$

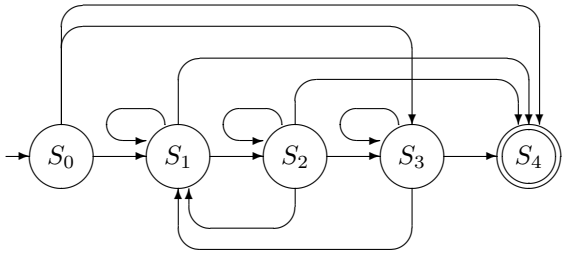
and so forth without end. While it is useful to consider the set of *all* scenario sequences that express desired behaviors, it is impractical to try to list all or even many of the system's infinite number of sequences. Instead, we seek to create a scenario network that expresses exactly those sequences that are allowable.

We construct this network by identifying its initial and terminal scenarios and each scenario's follow set, shown in the table below. An *initial scenario* is one that may begin a scenario sequence, as a *terminal scenario* may end one. A scenario's *follow set* is the set of scenarios that may follow it in a sequence.

Initial scenarios	S_0
Terminal scenarios	S_4

Scenario	Follow set
S_0	$\{S_1, S_3, S_4\}$
S_1	$\{S_1, S_2, S_4\}$
S_2	$\{S_1, S_2, S_3, S_4\}$
S_3	$\{S_1, S_3, S_4\}$
S_4	$\emptyset$

Based on the information in the table above, it is possible to produce a diagram for the corresponding scenario network, looking very much like a finite state machine with unlabelled arcs. We might consider that the first event of a scenario is the input that triggers a transition to that scenario; since every transition to a particular scenario shares the same trigger, we have not labelled the arcs in the diagram below.



This network permits all the scenario sequences listed above, and an infinite number of other desired sequences. Unfortunately, it also permits an infinite number of scenario sequences that are not desired, such as

– $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_2 \rightarrow S_4$
(caller left one message but subscriber listened to two)

Such undesired sequences must be ruled out, and we do this by assigning a precondition to each scenario. The precondition is required to be true in order for the scenario to begin. Each scenario is also assigned a postcondition which is guaranteed to be true at the end of the scenario. Thus at any point in a sequence within a network we can determine which scenarios can occur next by comparing their preconditions with the postcondition fulfilled by the scenario which has just finished.

The scenario network with pre- and postconditions added serves as an operational model that simulates the behavior of the system it specifies. The precise expression of a scenario's pre- and postconditions is useful, as these express important facts about the purpose of the scenario, what it requires before it can run, and what it accomplishes once completed. Our experience shows that we already have a rough idea of what a scenario needs and accomplishes before we actually write the conditions, so that the work is in part already done even if not made explicit. The formal nature of the pre- and postconditions offers the possibility of automated analyses of the system's behaviors.

Figure 1 presents the example EMS scenarios with pre- and postconditions added, and Figure 2 diagrams the corresponding scenario network with the pre- and postcondition for each scenario. A single primitive term is used to express the pre- and postconditions: the number of new messages n awaiting the subscriber. A primed variable (e.g. n') represents the value after the scenario, and a plain variable the value before the scenario.

The pre- and postconditions in Figure 1 suffice to express the desired connections between the scenarios, and to restrict the possible sequential paths to only those that are desired.

Fig. 1. Scenarios with conditions

S_0. System startup.

Precondition: $(false)$ (initial scenario only, precondition never satisfied)
Postcondition: $(n' = 0)$

1. Administrator starts up the system.

S_1. Caller leaves a message for subscriber

Precondition: $(true)$
Postcondition: $(n' = n + 1)$

1. Caller leaves a message and hangs up.

S_2. Subscriber listens to a new message.

Precondition: $(n \geq 1)$
Postcondition: $(n' = n - 1)$

1. Subscriber dials the *next new* command.
2. EMS plays his/her next new message.
3. EMS changes the message's state from "new" to "old".

S_3. Subscriber has no more new messages to listen to.

Precondition: $(n = 0)$
Postcondition: $(n' = 0)$

1. Subscriber dials the *next new* command.
2. EMS says he/she has no more new messages and hangs up.

S_4. System shutdown.

Precondition: $(true)$
Postcondition: (terminal, so no postcondition)

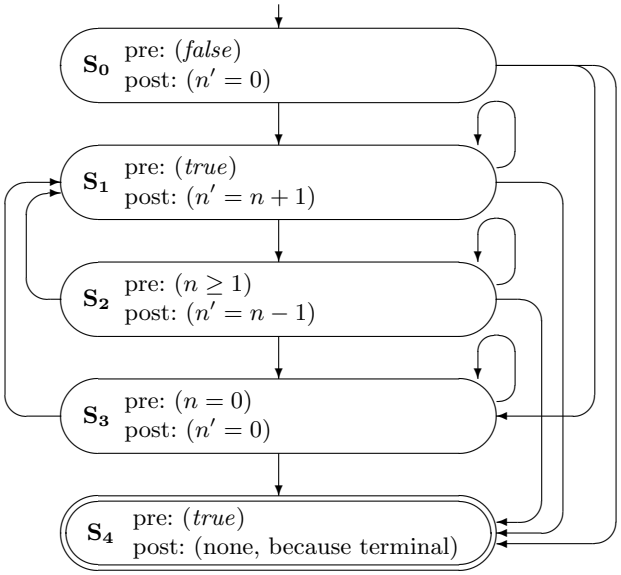
1. Administrator issues the shutdown command.
2. EMS stops accepting new calls.
3. EMS shuts down when all calls are completed.

Concurrency (of callers leaving messages, for example) is appropriate for the EMS. We extend the scenario sequences to *scenario multipaths* which may ramify and rejoin as they progress through the network. Further discussion of concurrency in scenario networks is beyond the scope of this paper.

4 EMS case study

The scenarios for the EMS were produced and reviewed over a period of approximately two calendar months by the two authors and a domain expert. All three are highly skilled in requirements specification and the use of scenarios, and two have extensive domain expertise. Two possess Ph.D.'s and the third is a Ph.D. candidate, with 30 or more years of industrial software development experience among them, and substantial specialized expertise in requirements engineering.

Fig. 2. Network diagram with conditions



We wrote a set of 42 requirements (see Appendix A) and constructed a set of 32 scenarios. In anticipation of the eventual scenario network, each scenario was annotated with pre- and postconditions as it was produced. The process was conducted as meticulously and thoroughly as possible. Inspections, detailed walk-throughs, and reviews were held weekly. As a result of what we perceived as a high-quality effort, we believed that the resulting scenario set was an unusually complete and consistent specification.

Once we had completed the process described above, the main author assembled the scenarios into a scenario network. The initial approach taken was to assume the pre- and postconditions were close to correct, and to connect the scenarios based on satisfaction of preconditions by postconditions. This approach proved unsatisfactory, in part because it required tedious work done by hand, but more importantly because the resulting network connected far too many scenarios and the narrative structure of the network was obscured. The large number of connections between scenarios at this stage of our analysis allowed sequences of scenarios that we knew should be ruled out; the postconditions were too strong compared to the preconditions, causing too many preconditions to be satisfied. We observed that the narrative had become an emergent property of the network rather than acting as the primary organizing principle, and in the presence of errors in pre- and postconditions the narrative was obscured and did not emerge. After two days of iteratively repairing approximately a quarter of the pre- and postconditions and reexamining the resulting network, we set this purely condition-centric approach aside and began again with a compositional

approach from a narrative perspective. This enabled us to view the scenario network as being recursively composed of successively smaller and smaller "stories".

Our compositional approach began with the consideration of the system-plus-environment's behavior as a single high-level story. At each stage in the approach, each story was decomposed further if it sensibly could be. Initially the single story was decomposed into a narrative for the system, a narrative for each system subscriber, and a narrative for each caller leaving a message. At the next stage, when the caller and subscriber narratives were decomposed into component scenarios, we quickly identified classes of equivalent scenarios (equivalent in the sense that in a scenario sequence, any of them in a class could be substituted for any of the others). This immediately indicated equivalence between pre- and postconditions, respectively, of scenarios in each class. The equivalence is reflected in corresponding connections in the scenario network. At this point, the scenario network was substantially complete in form. The concurrency inherent in the requirements became clear once the scenario network had been clarified; for example, a caller can leave a new message for a subscriber while that subscriber is checking his/her messages.

The technique of combining the scenarios into a network resulted in the discovery of two requirements errors and ten new scenarios describing behavior not covered or only partially covered by the original 32 scenarios. The discovery of so many errors and omissions showed us the value of scenario networks.

5 Lessons learned and future work

We learned several lessons during the course of the case study.

A purely condition-centric approach is not an effective strategy.

We found that initial concentration on scenario pre- and postconditions is not an effective means of constructing a scenario network, and tends to obscure the narrative structure of the system that we wish the scenario network to express.

Scenario networks help uncover missing scenarios

Constructing the scenario network for the EMS scenario set enabled us to discover ten new, previously overlooked scenarios. While analyzing the scenario network we realized that we were missing scenarios to describe system startup and shutdown, the termination of calls by a subscriber and by a caller leaving a message, and timing out a caller who does nothing. This resulted in the addition of five new scenarios. The scenario network also showed that we had not considered the terminal scenario for what happens when the system shuts down. Additionally, we found that four scenarios contained implicit alternative "exits" for which the main postcondition was inappropriate; these scenarios were split, resulting in the creation of four additional new scenarios.

Scenario networks help identify requirements errors

During scenario analysis, so many scenarios are typically generated that it becomes difficult to catch errors and inconsistencies systematically early on. The

scenario network enabled us to find both errors and inconsistencies that had not been identified during the initial period of careful reviews and walkthroughs. We discovered one requirements error by noting unexpected stub branches. A *stub branch* of a scenario network is a scenario that is not reachable, but is not a initial scenario; or a scenario that is not leavable, but is not a terminal scenario. Specifically, we observed that the scenario that presented archived messages to the subscriber lacked appropriate pre- and postconditions, and while investigating this we discovered that we had incorrectly separated the scenarios for presenting archived messages from the scenarios for presenting all other messages, and that the requirement that these scenarios traced back to was incorrect. We discovered an additional requirements error (in the specification of the sequence in which messages are presented) while tracing through the requirements related to the first incorrect one. Thus creation of the network uncovered two requirements errors.

Scenario networks express the sequences of scenarios

Scenario networks provide a richer and more expressive specification of a system's expected behaviors than a set of unconnected scenarios. Specifically, a scenario network forces analysts to consider the relationships and dependencies across all scenarios for a given system. A scenario network makes it possible to rule out disallowed sequences of events, so that analysts may focus on those sequences which are possible while ensuring that the system's constraints are represented in the pre- and post-conditions of each scenario. A scenario network also provides a way to express those aspects of a system's narrative structure that are too large-scale to appear in individual scenarios. Our customer had requested a menu tree to aid him in expressing the sequences in which behaviors would occur in the EMS; however, a tree is limited in that it only portrays branching. The scenario network enables us to represent not only simple branching, but also the various other ways in which scenarios may follow each other. A scenario network is thus far more expressive than a simple menu tree.

Areas of concern for scenario networks

In this case study the scenarios were relatively short, which made it easier to give each scenario a single exit and a single postcondition for it. On the other hand, the shorter scenarios made it more difficult to uncover the larger narrative structure of the system. We are examining these issues in our continuing research.

One of the concerns for scalability is the potential for an explosion in the number of system states as larger and more complex systems are considered. Our use of primitive conditions on system states, rather than direct consideration of the states themselves, mitigates the effect of such a state explosion; however, we will continue to examine this issue in our future work.

Some practitioners say they use gaps in the coverage of scenarios as a means of abstraction. They abstract from unimportant aspects of system behavior by not creating scenarios for it. Scenario networks require that abstraction be done instead by leaving the unimportant behavior out of the scenarios that are created. A related issue is the desire of some practitioners to use scenarios only as

examples, not as part of a description of the system's behavior. We believe that the creation of a scenario network does not make its component scenarios any less useful as examples, and that their usefulness is enhanced by relating them to each other and uniting them into a single specification.

Future work

We are conducting a larger case study of scenario networks to investigate the scalability of this technique and to extend its handling of concurrency.

Although we have not discussed goals in this paper, our current research is actively addressing the role of goals and scenarios in requirements, and we envision a clear mapping between goals and postconditions. We are investigating how scenario networks might act as an operational specification, and what results could be obtained from this approach. And we are in the process of formalizing our definitions of several semantic relationships between scenarios, such as precede- and follow-equivalence and a behavioral subtype relationship. We expect that our earlier work [1] which used syntactic relationships to identify dependencies, inconsistencies, and gaps in sets of scenarios, will advance using these semantic relationships which arise in scenario networks.

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. The authors also thank Klaus Pohl, Barbara Paech, and Karl Cox for their insightful comments at the REFSQ 2001 workshop.

Appendix A. Requirements for the EMS

As this set of requirements exhibits an interesting amount of complexity, a useful number of features some of which interact, several actors with different roles, and concurrency, yet is not too large, we present it to the RE community as a possible new problem to work with and on which to try techniques.

Definitions

announcement A subscriber's *announcement* is a recording that a caller hears when he or she reaches the subscriber's voice mailbox.

archived message An *archived message* is a message that the subscriber has listened to and has marked so that he/she can listen to it again later. An archived message can be kept for some long (but not unlimited) period of time.

caller A *caller* is a person (or automatic device) who has called a subscriber's voice mailbox.

command A *command* is an input to the EMS from a subscriber or caller. Commands are presently given by punching keys on a telephone keypad, although there are other possibilities.

EMS *EMS* is the Enhanced Voice Mail System.

erased message An *erased message* is one that has been erased. Nothing else can be done to it.

held message A *held message* is one that has been listened to or skipped, but not archived.

message state A *message state* is one of the following: new message, held message, archived message, or erased message. Each message is initially a new message.

new message A *new message* is one for which no action has yet been taken.
old archived message An *old archived message* is an archived message that was received more than some certain interval of time ago.
old held message An *old held message* is a held message that was received more than some certain interval of time ago. The interval of time need not be the same as that for old archived messages (and is expected to be substantially shorter).
passcode A *passcode* is a confidential sequence of digits, asterisks, and hashes (the symbols on a telephone keypad) that is used by a subscriber to identify himself/herself to the EMS.
private A *private message* is one that the caller that left it identified as such by a command; private messages may not to be forwarded.
recipient A *recipient* is either a subscriber (who is identified by his/her phone number), or a group of subscribers (the group is identified by a two-digit number). Ordinary telephone users who are not subscribers cannot be reached using the EMS features that require a recipient.
stuttered dial tone A *stuttered dial tone* begins with an alternating tone-silence pattern.
subscribed phone A *subscribed phone* is the phone of a subscriber; callers to this phone are directed to the subscriber's voice mailbox if their call is not answered after a certain number of rings (or if the subscriber's phone is busy).
subscriber A *subscriber* is a person for whom the EMS provides a voice mailbox.
urgent 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.

Requirements

R1. Access and Authentication
- **R1.1. Access EMS** A subscriber shall telephone the EMS in order to use it.
- **R1.2. Access from another phone** The EMS shall require a subscriber to enter his/her phone number when calling from a phone other than the subscribed phone.
  - **R1.2.1. Identity cue** The EMS shall announce the subscriber's name.
- **R1.3. Passcode** The EMS shall require a subscriber to dial a passcode.
- **R1.4. Passcode authentication** The EMS shall authenticate the dialed passcode.
- **R1.5. Subscription** Subscription to voice mail is not handled by the EMS.
  - **R1.5.1. Initial passcode** A subscriber shall be given an initial passcode by the system that handles subscription.

R2. Configuration Management
- **R2.1. Change passcode** The EMS shall allow a subscriber to change his/her passcode.
- **R2.2. Configure announcement** The EMS shall allow a subscriber to configure the announcement a caller hears before leaving a message.
  - **R2.2.1. Record announcement** The EMS shall record the subscriber's name and announcement.
  - **R2.2.2. Default announcement** The EMS shall allow a subscriber to choose a default announcement rather than recording one of his/her own.
  - **R2.2.3. Confirm announcement** The EMS shall play the subscriber's newly recorded name and announcement for confirmation. the EMS shall require the subscriber to confirm the name and announcement before saving it.
  - **R2.2.4. Announcement configuration exit** The EMS shall allow a subscriber to exit from configuring his/her announcement.
  - **R2.2.5. Configure announcement bypass** The EMS shall allow a subscriber to configure his/her announcement to allow or prevent callers from bypassing it.
- **R2.3. Phone number group** The EMS shall allow a subscriber to set up recipients that stand for groups of phone numbers.

R3. Subscriber interactions with the EMS
- **R3.1. Notification**
  - **R3.1.1. Check for messages** The EMS shall allow a subscriber to check for messages.
  - **R3.1.2. Number of messages** The EMS shall notify a subscriber of how many new, held, and archived messages he/she has when the subscriber checks.
  - **R3.1.3. Pager notification** The EMS shall support pager notification; that is, the EMS shall be able to call a subscriber's pager when an urgent message is received.
  - **R3.1.4. Stuttered dial tone** The EMS shall support notification by stuttered dial tone; that is, the EMS shall interact as necessary with other systems so that when a subscriber has one or more new messages, the subscribed phone will give a stuttered dial tone rather than a standard dial tone.
  - **R3.1.5. New message light** The EMS shall support notification by "new message" light; that is, the EMS shall interact as necessary with other systems so that if a subscribed phone has a "new message" light, that light is on when the subscriber has one or more new messages.
- **R3.2. Message processing by a subscriber**

R3.2.1. Presentation sequence The EMS shall present messages in the following order: first urgent new messages, then all other messages. Within these groupings, the EMS shall present messages in reverse chronological order (newest to oldest).
R3.2.2. Listen to messages The EMS shall allow a subscriber to enter a command and listen to messages one after another. Listening to a new message changes its state to "held message".
R3.2.3. Skip rest of message The EMS shall allow a subscriber to skip the remainder of a new, held, or archived message rather than listen to all of it. This is allowed for messages whether they are urgent, private, or neither. Skipping a new message changes its state to "held message". Skipping a held or archived message does not affect its state.
R3.2.4. Next message The EMS shall move to the next message when the subscriber takes one of the following actions on the current message: skip, archive, or erase.
R3.2.5. Jump within message The EMS shall allow a subscriber to skip around within a message while listening to it: to its beginning or end, and 5 seconds forward or back.
R3.2.6. Date and time After the EMS plays a message, the EMS shall also play the time and date the message was received and the phone number from which it came, if the subscriber requests it by a command.
R3.2.7. Erase message The EMS shall allow a subscriber to erase a new, held, or archived message. Erasing a message changes its message state to "erased message".
R3.2.8. Archive message The EMS shall allow a subscriber to archive a held message. Archiving a message changes its state to "archived message".
R3.2.9. Reply The EMS shall allow a subscriber to reply to a held or archived message.
R3.2.10. Forward The EMS shall allow a subscriber to forward a held or archived message to a recipient, if that message is not a private message.
R3.2.11. Forward with preface The EMS shall allow a subscriber to forward a held or archived message to a recipient, if that message is not a private message, and to record a preface to be forwarded along with the message.
R3.2.12. Record new The 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.
R3.2.13. Idle limit The EMS shall hang up if no action is taken within an administrator-determined limit of time.
R3.2.14. Prompt on old The EMS shall prompt a subscriber to erase or archive each old held or archived message before the EMS allows the subscriber to listen to any other messages or take any other action other than forwarding the old message.
R3.2.15. Erasing is final The EMS shall allow no actions on an erased message.

R4. Caller interactions with the EMS
R4.1. Play announcement The EMS shall play the subscriber's name and announcement.
R4.2. Leave message The EMS shall allow a caller to leave a message.
R4.3. Review message The EMS shall allow a caller to review the message he/she has just left, and re-record it if he/she wishes.
R4.4. Urgent or private The EMS shall allow a caller to distinguish his/her message as urgent or as private by giving a command.
R4.5. Return to operator The EMS shall support caller-return-to-operator (allow a caller to press 0 to reach a receptionist).
R4.6. Bypass announcement If the subscriber has configured his/her announcement to allow it, the EMS shall allow a caller to bypass the announcement by pressing a code.
R4.7. Call EMS directly The EMS shall allow a caller to leave a message for a subscriber in two ways: by calling the subscriber's unanswered number, and by calling the EMS directly.

Appendix B. List of Scenarios for the EMS

This is a list of the scenarios obtained after creating and analyzing the scenario network. Due to space limitations, the scenario network itself is left as an exercise for the reader. Send your solutions to the authors; the first entrant whose network matches ours will receive a lifetime supply of pre- and postconditions.

S0. EMS startup
S1. EMS shutdown
S2. Subscriber calls EMS
S3. Subscriber calls EMS from an unsubscribed telephone
S4. Subscriber calls EMS from some other subscriber's telephone
S5. Subscriber changes his/her passcode

S6. Subscriber configures his/her announcement
S7. Subscriber sets up a group of phone numbers as a recipient
S8. Subscriber checks for new messages
S9. New message notification by stuttered dial tone
S10. New message notification, by indicator
S11. No new message notification, by indicator
S12. Subscriber listens to a message
S13. Subscriber has no more messages to listen to
S14. Subscriber skips to next message
S15. Subscriber skips but has no more messages
S16. Subscriber skips around in a message while listening to it
S17. Subscriber listens to the time a message was received
S18. Subscriber erases a message
S19. Subscriber erases the last message
S20. Subscriber archives a message
S21. Subscriber archives the last message
S22. Subscriber forwards a message
S23. Subscriber forwards a message with a preface
S24. Subscriber calls the person that left a message
S25. Subscriber records a message and sends it to someone
S26. Subscriber takes no action for a long time
S27. Subscriber is prompted about old held messages
S28. Subscriber is prompted about old archived messages
S29. Subscriber disconnects from EMS
S30. Caller calls subscriber and leaves a message
S31. Caller reviews his/her message
S32. Caller reviews and re-records his/her message
S33. Caller distinguishes his/her message as urgent
S34. Caller distinguishes his/her message as private
S35. Caller decides he/she needs to speak to a receptionist
S36. Caller doesn't want to listen to the subscriber's announcement
S37. Caller calls EMS directly and leaves a message
S38. Caller takes no action for a long time
S39. Caller disconnects from EMS

References

1. T. Alspaugh, A. 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. 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.
3. R. Buhr and R. Casselman. *Use case maps for object-oriented systems.* Prentice Hall, 1996.
4. 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.
5. 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.
6. 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.
7. 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.