Fifth International Workshop on Comparative Evaluation in Requirements Engineering (CERE'07), pages 1–10, doi:10.1109/CERE.2007.3

Clarity for Stakeholders: Empirical Evaluation of ScenarioML, Use Cases, and Sequence Diagrams

Thomas A. Alspaugh Susan Elliott Sim Kristina Winbladh Mamadou H. Diallo
Leila Naslavsky Hadar Ziv Debra J. Richardson

Department of Informatics
Bren School of Information and Computer Sciences
University of California, Irvine
{alspaugh,ses,awinblad,mamadoud,lnaslavs,ziv,djr}@ics.uci.edu

Abstract

We studied the clarity of three requirements forms, operationalized as ease of problem detection, freedom from obstructions to understanding, and understandability by a variety of stakeholders. A set of use cases for an industrial system was translated into ScenarioML scenarios and into sequence diagrams; problems identified during each translation were noted; and all three forms were presented to a range of system stakeholders, who were interviewed before and after performing tasks using the forms. The data was analyzed, and convergent results were triangulated across data sources and methods. The data indicated that ScenarioML scenarios best support requirements clarity, then sequence diagrams but only for stakeholders experienced with them, and finally use cases as the least clear form.

That I require a clearness ...
To leave no rubs nor botches in the work ...

Shakespeare, *The Tragedy of Macbeth*, III.i.

1. Introduction

High quality requirements are critical to software project success. Research over the past quarter century has shown that requirements errors rapidly become more expensive to address, and their consequences if not addressed can include a crippled product or cancelled development. Incomplete requirements, lack of sufficient stakeholder involvement, and systems that fail to meet stakeholder needs are all too common [4, 5, 8, 13]. These issues have led us to consider *clarity for stakeholders* as a fundamental prerequisite of requirements quality. In this paper, we report an empirical study evaluating the clarity of three requirements specification formats (use cases, sequence diagrams, and the first author's ScenarioML). We operationalize clarity through the following research questions:

1. Which format most readily permits the detection of problems in requirements?
2. Which format's structure least conceals errors?
3. Which format do stakeholders find most clear?

The study has two phases, analogous to first testing the transmission characteristics of three different telephone cables, before testing the ability of specific humans to hear conversation across the cables. In the first phase (Figure 1), we examine how well the three formats represent the same requirements specification. We obtain three comparable requirements specifications by translating an existing set of industrial use cases into ScenarioML scenarios and into sequence diagrams, and then independently re-translate selected scenarios and sequence diagrams back to use cases for comparison. The use cases were for the prototype of the Mirth healthcare middleware system [11] developed for our industrial partner WebReach, Inc. [15]. We analytically evaluate the three formats by comparing the number and kind of problems that arise or are identified during each translation. In the second phase (Figure 2), we ask three stakeholders from WebReach to complete a test of their comprehension of material presented in each of the requirements specification formats, with pre- and post-interviews to elicit reactions, opinions, and insights. We use a Latin square distribution of stakeholders, tasks, and specifications (Figure 2) in order to obtain effective cross-participant and cross-format comparisons. We evaluate each stakeholder's task performance and then code (i.e. chunk and categorize) the interview transcripts, which are qualitatively analyzed to identify themes [7, 9].

We chose this mixed-methods approach [7] for several reasons. Our goals were to explore the concept of clarity for

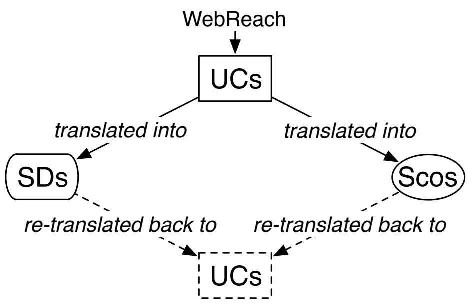


Figure 1. Translation to obtain three formats

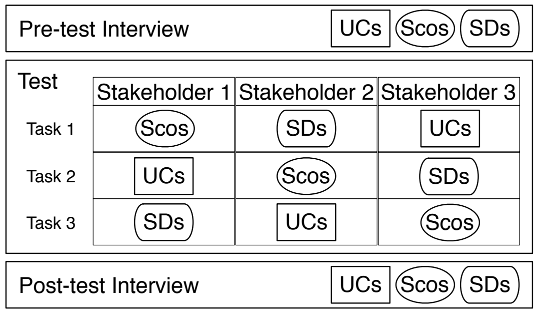


Figure 2. Stakeholder interviews and test

stakeholders in the context of different requirements specification formats, to validate the clarity of ScenarioML specifically, and to gain insights into the views, beliefs, preferences, activities, and goals of various kinds of stakeholders in their use of requirements specifications. The existence of public use cases for the Mirth system, and our connection with stakeholders at WebReach, Inc. provided an opportunity for in-depth study using industrial use cases not developed by us and interactions with a range of industrial stakeholders. The context of the study permitted collection of both quantitative and qualitative data concurrently, enabling us to obtain focused data for addressing specific questions and also more-open-ended data to provide insight and understanding and guide future research. Finally, the approach supported collection of data from multiple sources and triangulation across these data sources (Figure 3), thus providing cross-correlation and corroboration in the event that results from different sources converged, as they did in this case [7]. While this research design does not produce statistically generalizable results, it does provide insights into the properties of requirements specification formats, which was our main goal.

The data from both phases of the study indicated that the ScenarioML format was the most clear, followed by sequence diagrams (but only for those familiar with them), followed by use cases. In general, the researchers who performed the translations were able to detect more problems, such as inconsistency, when translating use cases into ScenarioML than when translating them into sequence diagrams. While it was easier to detect high-level problems with ScenarioML, it was easier to detect problems with low-level details with sequence diagrams. This both confirmed and delineated the kinds of clarity provided by ScenarioML. Stakeholders also preferred ScenarioML scenarios over use cases, and usually over sequence diagrams. They found ScenarioML to be clearer and expressed a preference for this format, except in certain specific situations, such as communication among different architectural components.

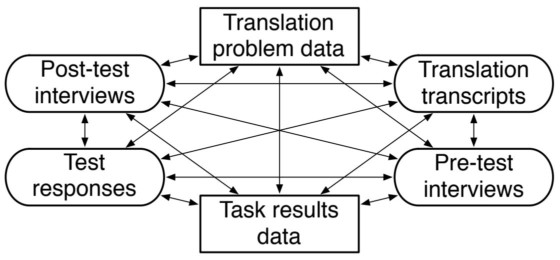


Figure 3. Triangulation across data sources

The remainder of the paper is organized as follows. Section 2 surveys related work. Section 3 presents the study's three requirements specification formats. Section 4 discusses the method used. Section 5 presents study results, which are discussed in Section 6. Section 7 lists lessons learned, and Section 8 concludes and outlines future work.

2. Related Work

Requirements clarity and related concepts have been noted by a number of researchers. Among the most focal are the following. Heninger proposed operational objectives closely related to our research questions [10]. Ben Achour *et al.* note the advantages of narrative text use cases for stakeholders, and discuss guidelines for improving their quality which would also address clarity [3]. Sommerville lists lack of clarity as one of the top problems for natural language requirements [12].

There has been little investigation of clarity as a quality attribute of requirements. We know of only one contribution that focuses specifically on clarity, Wasson's CLEAR (Cognitive Linguistic Elicitation and Representation) method for writing requirements and a case study in which CLEAR was applied to the requirements for a medical device [14]. In contrast to CLEAR which focuses specifically on the wording of requirements, we examine how requirements specification formats support or obstruct clarity.

Williams *et al.* identify several weaknesses of use cases leading to confusion in their practical use, including weak semantics and mismatches between text-based use cases and the UML metamodel of use cases [17]. Whittle proposes a more formal modeling paradigm for use cases, called use case charts, strongly influenced by UML semantics [16]. Both Whittle and Williams *et al.* seek to increase use-case-based automation, for example for test case generation, validation, or simulation. We distinguish clarity for stakeholders from formality, noting that formality provides (among other benefits) focus, compactness, and precision, but may not be appropriate for most stakeholders, who often are not trained in formal approaches.

3. Requirements Specification Formats

3.1. Use Cases

A use case is a description of a sequence of interactions between the system and one or more external actors that results in an outcome of value to at least one actor. A use case "describes the system's behaviors and interactions under various conditions as a response to a request on behalf of one of the stakeholders—the primary actor—showing how the primary actor's goal gets delivered or fails" [6].

Use cases are a structured text form, typically written using a use case template whose form may vary widely from project to project. The use case author fills in text fields such as use case name, goal, brief description, pre- and post-conditions, and normal, alternative, and exceptional flows. Thus a use case typically contains several scenarios, providing a primary path to reach the stated goal and possibly alternative paths to the goal or paths describing ways the goal can fail to be met. The Mirth use cases correspond to Cockburn's "casual" use case structure, customized as he recommends to the project and its situation [6].

3.2. Sequence Diagrams

A sequence diagram is a UML diagram that documents a sequence of object interactions that take place for a path through use case. They are typically used during analysis and design activities but, much like use cases, have been used at many different levels of detail and abstraction, even as replacement for and in lieu of use cases. They are well suited for illustrating the interplay between several architectural or design components, or between several actors and a system, and show the interactions in terms of message passing. The entities involved in the interactions are represented as vertical timelines, and messages or operations can be followed in sequence on each timeline from top to bottom.

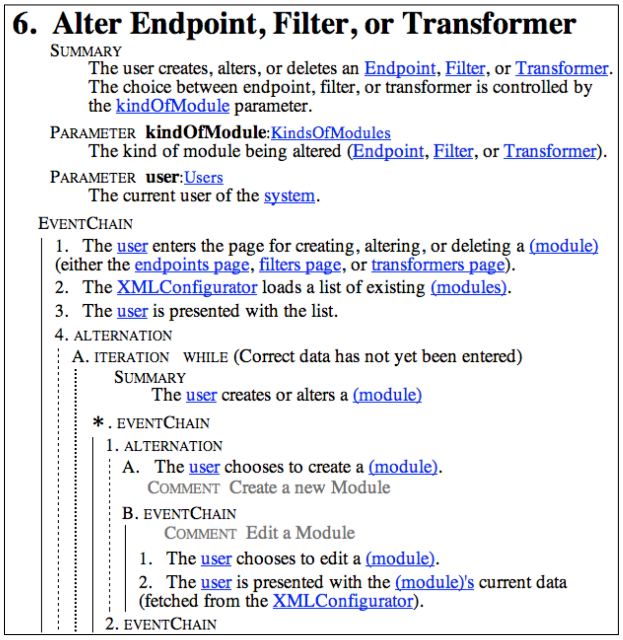
3.3. ScenarioML

ScenarioML is a language for scenarios, designed to be read and written by people while also accommodating automated processing. It is an XML language that is presented to stakeholders through an automatically-produced HTML representation, or by an Eclipse editor plugin currently in development. The goals of ScenarioML are to make scenarios that are clearer, more useful, and more effective. The scenarios in this study are represented using a combination of *recursively-defined events*, *scenario parameters*, *references*, and *ontologies*.

The basis of recursively-defined events are *simple events*, each of which describes in words one thing happening in the world. A *compound event* groups several subevents to describe several things happening in the world in a temporal pattern, either an event chain (the only compound events appearing in the study scenarios), directed acyclic graph, or general graph. An *event schema* compactly expresses a set of related combinations of events, such as an iteration representing any of several chains of repetitions of its subevent, or an alternation representing any one of a group of alternative subevents. Finally, an *episode* uses an entire scenario as one event of another scenario, with arguments for the reused scenario's parameters to specialize it for the context in question. These recursively-defined events support clarity for stakeholders by providing expressive means of stating how events unfold over time, and by supporting refactorings among equivalent or specialization-related event patterns by which a stakeholder can verify what is expressed. Figure 4 shows an excerpt from one of the study scenarios, presented in HTML and illustrating several of these kinds of events.

ScenarioML supports explicit references to entities and terms defined elsewhere. In the XML, references are expressed with markup elements in the text of a simple event. In the HTML presentation, a reference appears as a hyperlink to the referent. References support clarity by allowing scenario authors to link each referring word or phrase to its referent or definition, so that a reader can learn or confirm what was intended. The referents may be entities or types in the ontology of the world described by the scenarios, or parameters of the scenario containing the event with the reference. ScenarioML also supports references to entities newly created or identified during the course of a scenario's events. This is particularly useful in a scenario involving two or more of the same kind of entity. An example in point from the current study is the scenario "Add/Edit/Delete Users", in which the actor (who is a user) can add a new user (not himself), edit a user (including himself), or delete a user (but not himself). There are a number of references to a 'user' in this scenario, but not all to the same user. ScenarioML references and new-instance markup allow this to be done unambiguously.

ScenarioML has been the basis of work with autonomous animated social agents, including a novel visualization of software scenarios as social interactions between au-



6. Alter Endpoint, Filter, or Transformer

SUMMARY
The user creates, alters, or deletes an Endpoint, Filter, or Transformer. The choice between endpoint, filter, or transformer is controlled by the kindOfModule parameter.

PARAMETER **kindOfModule**:KindsOfModules
The kind of module being altered (Endpoint, Filter, or Transformer).

PARAMETER **user**:Users
The current user of the system.

EVENTCHAIN
1. The user enters the page for creating, altering, or deleting a (module) (either the endpoints page, filters page, or transformers page).
2. The XMLConfigurator loads a list of existing (modules).
3. The user is presented with the list.
4. ALTERNATION
   A. ITERATION WHILE (Correct data has not yet been entered)
      SUMMARY
      The user creates or alters a (module)
      *. EVENTCHAIN
      1. ALTERNATION
         A. The user chooses to create a (module).
            COMMENT Create a new Module
         B. EVENTCHAIN
            COMMENT Edit a Module
            1. The user chooses to edit a (module).
            2. The user is presented with the (module)'s current data (fetched from the XMLConfigurator).
      2. EVENTCHAIN

Figure 4. ScenarioML scenario in HTML presentation oriented to stakeholders (excerpt)

tonomous animated agents generated automatically from the ScenarioML [1]. ScenarioML has supported an approach for goal-driven specification-based testing, in which the lower-level goals for a system are organized into plans for achieving higher-level goals against which the system is tested [19], and is the basis for our requirements-based testing approach, in which a system is driven by and checked against its requirements scenarios [18].

4. Method

Our goal is to evaluate the clarity for stakeholders of three requirements forms. We apply the Goal Question Metric approach [2] to produce a measurement model operationalizing our goal into research questions, and associating each question with data that can be evaluated (Figure 5). We begin by summarizing definitions of clarity from authoritative dictionaries as "easy to understand, fully intelligible, free from obscurity, unmistakable," and derive three research questions.

1. Which format most readily permits the detection of problems in requirements?

2. Which format's structure least conceals errors?

3. Which format do stakeholders find most clear?

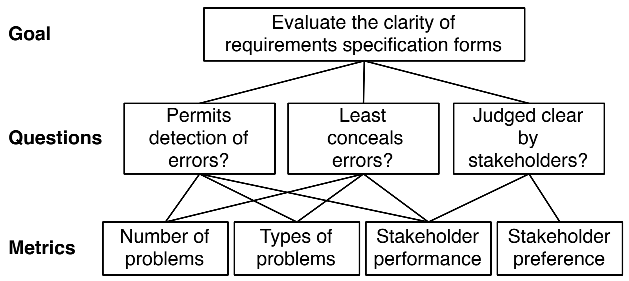


Figure 5. Goal Question Metric model

We decomposed the high-level goal into these three research questions, so that we could triangulate the phenomenon using data about user behavior, format properties, and stated user preferences. The first research question seeks to determine which format allows users to find more errors in the requirements. The second question probes the underlying, or structural, properties of the format. The final question addresses subjective experiences.

To answer these questions, we conducted a two-part empirical study, with the first focusing on the capabilities of the formats and the second focusing on stakeholder performance using the formats. It was necessary to evaluate these two aspects separately, because we needed to factor out the capabilities of the formats in terms of their carrying capacity, in order to isolate human behavior with the formats.

In this study, we created and studied the requirements for a single system, Mirth, expressed in three different formats, use cases, sequence diagrams, and ScenarioML. In the first part, we examined the clarity of the formats. The metrics that we collected were the number and types of problems encountered when creating the documents. In the second part, we examined the clarity of the formats when being used by stakeholders. The metrics that we collected were their subjective statements on their experiences with the documents and their objective performance on tasks testing their understanding of the documents presented in each format.

4.1. Subject System

We studied use cases for WebReach's Mirth system [11]. Mirth is an open source cross-platform interface engine for transferring healthcare information between systems that use different data formats. There were ten use cases ranging in size from 2 to 20 steps (average 7.3), and with zero to six alternate flows per use case (average 1.9). The use cases had been developed by a team of senior undergraduate software development students working with WebReach stakeholders, and then used as requirements for prototype development.

4.2. Requirements Representation

In the first phase of the study, we investigated the clarity of the three requirements specification formats.

We chose ScenarioML because we wanted to conduct a formative evaluation of it, and it is similar in structure to use cases, but with additional features intended to support clarity. In this sense, ScenarioML scenarios are more complex than use cases, but result in documents that are more straightforward, because event chains are linear.

We chose use cases because they are similar in some ways to ScenarioML scenarios and are widely used in industrial practice, and because of the availability and appropriateness of the Mirth use cases as a basis for the study.

We chose sequence diagrams because they are also widely used, and have distinctive similarities and differences with respect to use cases and scenarios that could result in interesting insights. All three formats describe sequences of events, but sequence diagrams do so graphically while use cases and scenarios are primarily textual.

4.2.1 Procedure

We translated the Mirth use cases into the other two formats, ScenarioML and sequence diagrams. Each translation was done by a different graduate student; both students were comparably proficient with use cases, and comparably proficient with their target formats (ScenarioML or sequence diagrams). Another researcher logged the translators' think-aloud verbal protocols of the translation process, including any problems or challenges that were encountered. A third graduate student analyzed the logs and the resulting requirements specifications for problems identified in the use cases during the translations. He characterized the types of problems by inducing categories from the data, as is done in a Grounded Theory approach [9].

Yet another researcher familiar with all three formats retranslated a sample of the scenarios and sequence diagrams back into use cases, with another researcher logging the retranslator's introspective narration of the process. Problems and insights from this process were correlated where possible with those obtained in the first translations. We did this to cross validate the translations that were performed and to evaluate the process of translating requirements into use cases. This step allowed us to verify that the initial translations were done correctly, and that any information loss could be attributed to the specification formats.

4.3 Stakeholder Evaluation

In the second part of the study, we presented the three formats for evaluation by stakeholders. They were asked about their impressions of the formats in an interview and then tested on their ability to perform tasks with them. We achieved a high level of realism, though not necessarily generalizability, by using a subject system from industry and selecting participants, also from industry, who had prior experience with the subject system. Due to our stringent selection criteria for subjects, we were able to obtain a sample size of only four (one in the pilot study, and three in the main study). We compensated for this small number by using a Latin square design to ensure that each subject saw each problem and each format. This approach ensured scientifically valid qualitative results (although not statistically generalizable), that were highly realistic and provided novel insights into the underlying research problem.

4.3.1 Procedure

Each interview was conducted individually and lasted for about one hour. We began by asking participants about their impressions and experiences with the formats. We then gave each participant a short test to assess the clarity of the three formats for that participant. The test consisted of three tasks each involving a different format and a different scenario to probe their understanding. The tasks were chosen to highlight potential problems in clarity by revealing whether the participants fully grasped complex interactions, understood potentially confusing material, and detected errors.

Problem 1: Add/Edit/Clone/Delete a Channel This use case was complex, with challenging inter-relations between the main success scenario and alternate scenarios, and several needed pieces of information missing. The corresponding scenario and sequence diagram represented the complexity in the structures provided by those formats. A participant who completely understood what was presented by the assigned format would be able to paraphrase the steps involved and identify missing information.

Problem 2: Deleting a User This use case contained an alternate flow that did not fit into the main flow, and the use case title listed more functions than the steps of the use case described. Participants were asked to describe the sequence of steps necessary to delete a user. A participant who completely understood would be able to point out the ways in which the format did not tell them what was needed.

Problem 3: Change Channel Status Although the use case describes enabling a channel, and purports to describe disabling one, it does not provide all the steps needed for disabling. A participant who completely understood would notice this and be able to point out where steps are missing.

With three formats, three tasks, and three participants, we used a Latin square design. Each participant saw each format and each task but in different combinations, so that we could compare between—and within—each participant's performance. Figure 2 depicts the distribution to each participant of the tasks and formats.

The interviews were transcribed verbatim, without imposing sentence structure or clarification. We then coded and categorized utterances in the transcripts [7, 9]. We used 16 categories derived from the data such as preferences, familiarity, and hostility with regard to each format. During analysis, we identified statements that pertained to the clarity of each of the formats. We were careful to consider both positive and negative statements when summarizing the interviews and selecting examples.

4.3.2 Participants

Four stakeholders at WebReach of varied background and focus were participants in this study. The first in the list was a pilot subject and the remainder served in the main study.

Stakeholder 1 An enterprise software architect with over 20 years of experience, who has designed and implemented solutions for clients in a wide variety of industries and technologies. This participant's results were used as a pilot evaluation for the study's second phase.

Stakeholder 2 A manager with 20 years of software engineering and management experience, whose primary focus is business development and key-client account management.

Stakeholder 3 A software engineer who helped lead the architecture and development of the Mirth Project, and is currently a PhD student.

Stakeholder 4 A Mirth developer and an author of the original use cases, whose current focus is business development and project management.

4.4. Validity

In this subsection, we discuss the internal and external validity of the study, as well as threats to validity.

Internal Validity. Internal validity is the soundness of the relationships within a study. In the requirements translation portion of the study, we linked the clarity of a format with the clarity of a document produced using that format. These in turn, were measured indirectly in terms of the problems that the format could reveal. A requirements specification format demands inclusion of certain kinds of information, and presents certain kinds of information more prominently, and different formats have different demands and different presentations. Therefore, translating requirements from one format to another will reveal shortcomings of the source document and the source format.

External Validity. External validity is the degree to which the results from the study can be generalized. In this study, we used only one subject system, three students, and three stakeholders. This is a small sample size on all counts. However, we are not trying to create statistically significant results that can be generalized to a population of subject systems, students, or stakeholders. Rather, we are seeking to build theories and deepen understanding of requirements specification formats. In the requirements creation part of the study, we seek to reason analytically about properties and qualities of the formats. These properties will hold across subject systems and documents. In terms of stakeholders, the data are not as strong, so we use these only as corroborating evidence. However, we feel this evidence is valid, because it comes from actual stakeholders.

We are using qualitative and quantitative data from multiple sources regarding both behavior and opinion. This wide variety of data allows us to triangulate a phenomenon, the importance of requirements clarity. Because the data were produced in different ways and show different viewpoints on the issues, it is unlikely that they will converge due to chance. If the four metrics converge, the weight of their support will be magnified.

Threats to Validity. The students who participated in this study are authors on this paper and involved in some way in the design of ScenarioML. This may result in a bias in favor of ScenarioML. We attempted to minimize this threat by having the students work independently and evaluated their work independently.

5. Results

In this section, we present our findings regarding our four metrics. These metrics will be used in Section 6 to answer the research questions.

5.1. Number and Types of Problems Found

Following the translation exercise, the documents and logs were analyzed independently by another student. A catalog of the problems found was created. These problems were then grouped according to similarities. Based on the grouping, a taxonomy was created so that each problem encountered fit into only one category. The result was a comparison table (Table 1) between the problems found during the two translation activities. This table was verified by the translators to ensure that the taxonomy was accurate.

Category 1 in Table 1 addresses inconsistencies between the use case diagram and the textual use cases. Some use cases seemed to be represented in both, but under different names such as "CreateModifyModule" and "CreateModifyViewModule". Others were visible in the diagram and simply missing in the textual representation. Another problem that surfaced during the translation was that use cases do not clearly express iterations, this type of problem is

Problem categories	Found in SDs	Found in scenarios
1. UC diagram - textual UC mismatch	0	4
2. Iteration was not specified	0	2
3. Redundancy in the UCs	0	1
4. UC numbering scheme does not show where alternative flows start/stop	3	4
5. All possible alternatives not specified	0	2
6. Mistakes in the UC	1	4
7. Incomplete UCs	3	3
8. Two names referring to the same term	5	5
9. Problems with details of events	6	3

Table 1. Problems found during translation

shown in category 2. Category 3 shows structural problems, where clarity could have been gained by combining similar use cases. For example, three of the use cases perform exactly the same steps with different actors. Another problem was the confusing numbering of events that resulted in ambiguous alternatives and exception scenarios. Category 4 shows such problems, which were found in three of the ten use cases. Category 5 shows problems of missing alternatives. The option of deleting a user, for example, was listed as an option but there were no events that described this process. Category 6 shows mistakes, such as mixing up the order of events in the use cases. Category 7 addresses the problem of incompleteness, such as referencing non existing use cases, while category 8 deals with inconsistencies such using different names for the same actors. The last row combines problems in individual events, such as where data can be stored.

The data shows that ScenarioML revealed more problems in the original use cases than the sequence diagrams. However, sequence diagrams were better at showing problems with the details of events. In contrast, ScenarioML scenarios more effectively show problems at a high level of detail and interstitial problems, i.e., problems between use cases or between different parts of use cases.

5.2. Stakeholder Preference

Each participant was interviewed individually and asked about their preferences regarding the three formats.

In general, the participants preferred ScenarioML over the other two formats. While they pointed out a few things that were unclear and not intuitive in the scenarios, such as the star that represents a number of iterations, they did not perceive these as off-putting or think that other stakeholders would. However, they also thought that use cases might be superior for initial requirements gathering and for users to write themselves. Sequence diagrams were disliked by the two participants who were not already familiar with the notation. Sequence diagrams were seen as useful or advantageous in more restricted contexts and audiences. The participants liked ScenarioML although they had the same or less experience with this notation. The participants liked use cases but found ScenarioML scenarios to be an improvement in a number of ways. Below is a summary of the results per format.

Use Cases. The participants perceived use cases as “fairly easy to understand”, in particular the steps described in the use cases, the summaries of what the use cases do, and the main and alternative scenarios that express alternative flows. Things that the participants found difficult to understand included the numbering scheme for alternative scenarios, and the terms used in events. In terms of preference, the participants liked that the use cases were easy to read and that the notation was familiar. Some things the participants disliked were the lack of detail, the numbering scheme, the inconsistency in terminology, and the lack of hyperlinks to supporting documents.

Sequence Diagrams. One participant found the notation “pretty easy”, and understood the calls between components and the separation of layers. Two of the participants perceived the notation as confusing, in particular how to follow branches and options. The subject that understood the sequence diagrams liked them because of their graphical nature. Some things that the other two participants did not like were the lack of readability, the visual overload, and the lack of hyperlinks to supporting documents.

ScenarioML Scenarios. The participants found the notation easy to understand. The participants understood the concepts of parameters, ontology definitions, event chains, alternations, references, and instances. The participants liked that the notation was simple and textual with indentations, the declarative statements with parameters, that there were better and more consistent definitions of terms, that it was more rigid, and had hyperlinks to supporting documents and terminology.

The overall observation is that the participants think ScenarioML specifies scenarios with greater clarity, except in showing interactions with various layers of a system with a layered architecture.

5.3. Stakeholder Performance

Overall, the subjects exhibited low performance in finding problems in the use cases and sequence diagrams despite their familiarity with the subject system. We had expected better performance because we selected subjects who had worked with Mirth. Another unexpected result was the conflict between attitudes and performance on use cases. All three participants had positive attitudes towards use cases and thought they were clear. However, they had difficulty on the comprehension tasks involving use cases.

This comprehension difficulty implies that they were difficult to use because they were unclear, contradicting the initial reaction from the stakeholders.

In the remainder of this subsection, the performance on the three formats will be discussed in detail.

Use Cases. Overall the participants performed poorly when trying to identify problems in the use cases (see 4.3.1 for the task descriptions). In Task 1, the participant with the use case failed to identify any problems, although the alternate extension did not fit into the main use case and was not appropriate for the "Clone" option. However, the participant did note that "the numbering scheme for the steps can be confusing."

In Task 2, the participant with the use case listed a sequence of steps that did not make sense, since the option was missing in the use case. The participant concluded that the use case was "kind of ugly". The participant had not noticed this problem before, and did not think that the notation did anything to point out this problem.

In Task 3, the participant with the use case thought it was difficult to determine the starting point of the two scenarios (main and alternate), because they were different. The participant did not notice this problem when looking through the use cases during the interview.

Sequence Diagrams. In Task 1, the participant with the sequence diagram understood the notation well, and was positive toward it although no problems were identified.

In Task 2, the participant was asked to list the steps that delete a Mirth user, although the sequence diagram only contained steps for the disallowed case of a user deleting himself. The user gave a plausible sequence of steps inferred from the diagram, but did not realize his error. Despite this, the participant displayed a positive attitude towards sequence diagrams, stating "This specification makes it relatively clear to spot problems with the task."

In Task 3, the participant could not list the steps to disable a channel, saying "The form is a bit incomprehensible."

ScenarioML Scenarios. In Task 1, the participant with the ScenarioML scenario understood the scenario well.

In Task 2, the participant with the ScenarioML scenario understood the scenario well, and pointed out that the nested `EventChain` highlights the fact that something is occurring in an otherwise flat sequence of events and draws attention to it, and that "This causes me to further inspect these nested events more closely as they often seem to be problematic areas."

In Task 3, the participant with the ScenarioML scenario was able to determine the steps to disable a channel (which the scenario presented correctly), but his attitude conflicted with this successful performance; he stated "I can not really identify potential problems (as easily as the sequence diagram)", although in fact he had previously been unable to identify problems in a sequence diagram.

6. Discussion

This section is organized according to our three research questions. However, data from each metric may be used to answer more than one question. We also add qualitative data to help corroborate our conclusions, and add richness and immediacy to the discussions.

6.1. Which most readily permits the detection of problems in requirements?

In our study, the translation process is used to compare the ease of problem detection in each of our target formats. This comparison is based on the reasoning that information required by a target format, but missing from the source format, is indicative of a shortcoming in the source format.

We decide the strengths and weaknesses of a format by comparing numbers and types of problems identified in the translation to that format. A drawback of this approach is it does not permit a side-by-side comparison of use cases to either ScenarioML scenarios or sequence diagrams, as use cases were the source format in both translations. This mismatch is compensated for in part by the re-translation from these formats back to use cases.

However, data from the interviews and comprehension test shows that stakeholders detected few of the problems present in either the use cases or sequence diagrams they examined, which is consistent with the inference that use cases are not markedly better than sequence diagrams for uncovering problems.

The list of problems uncovered (see Table 1) support the following inferences: that sequence diagrams more readily detect problems in the details of events; and that ScenarioML scenarios more readily permit detection of larger-scale problems related to clarity of requirements. Overall, 3 of the problems found in the translations were uncovered only using sequence diagrams, 15 problems were uncovered by both formats, and 13 additional problems were uncovered solely by ScenarioML scenarios. The data from the interviews provided evidence converging with that from comparing the lists of problems uncovered during translation, indicating stakeholders thought sequence diagrams more effective for design tasks in which layers and architectural components are important, while ScenarioML scenarios would be more effective for requirements, as a contract between stakeholder and developer, and for high-level communication. This triangulation across two quite different sources of data strengthen the inference drawn from each.

Overall, we infer that ScenarioML scenarios more readily permit the detection of errors than sequence diagrams to a substantial degree, but that sequence diagrams appear to more readily permit detection of detailed problems within individual events. Use cases fall in between the two and seem to be more similar to sequence diagrams.

6.2. Which least conceals errors?

The degree to which the three formats obstruct understanding is addressed by the notes from the translation process, the comprehension tests, and the interviews. The student who translated use cases to ScenarioML scenarios commented that she frequently had to consult other documents for missing information. The comprehension test showed that all three stakeholders found the use case form obstructed understanding, with comments such as "it's not too helpful for me in understanding these types of issues at all", "doesn't do anything to point it [the problem] out", and "more difficult to spot what is a problematic step". The stakeholder who was familiar with sequence diagrams did not find any of the problems they contained (although he felt they were easy to understand). The stakeholders who examined ScenarioML scenarios indicated the form did not obstruct their understanding, and noted ways in which they felt the form may have helped. The interview data revealed essentially the same pattern: all stakeholders identified ways in which use cases obstructed their understanding; stakeholders' responses for sequence diagrams were divided by whether they understood the form or not; and all stakeholders commented on ways in which ScenarioML enhanced their understanding,while identifying two minor ways in which it obstructed their understanding. (The stakeholders did not understand that an iterated event numbered with $*$ indicated it may occur many times; and they wished they could optionally collapse compound events for clarity when looking at the overall picture.)

All three sources of data converged on the inferences that use cases most obstructed understanding, sequence diagrams did not obstruct understanding if one already understood the format, and ScenarioML tended to aid understanding rather than obstructing it.

6.3. Which do stakeholders find most clear?

The tests and interviews were the sources of data for this question. Overall, the stakeholders indicated they found ScenarioML scenarios clear. In contrast, the stakeholders made conflicting statements about use cases, and only stakeholders who had expertise in sequence diagrams liked them.

While all three stakeholders indicated that they felt they understood the original use cases, this is contradicted by their performance on the tasks, and noncommittal and sometimes contradictory comments such as "it's hard to tell" and "That's not that clear", but also "pretty straightforward" and "they're easy to read". Other statements also indicated their ambivalence towards use cases. For example, they said "this form of specification is confusing to me" and "perhaps this is a problem with the user creating the use cases versus the format of the use case itself".

Comments about sequence diagrams were also mixed. Stakeholders who had expertise in sequence diagrams found them to be clear; those who did not disliked them strongly and one stakeholder commented that if he had to create sequence diagrams, "I would be regretting the day I had to come in and do these".

In contrast, all three stakeholders found ScenarioML scenarios easy to understand, with comments such as "Very easy", "unambiguous", "this is a very good form for a developer to get something in", "a richer, more defined way of doing use cases", "wow, absolutely, I think that's a definite improvement" (over use cases), "That's good, I like that", and "It's kind of like a UseCase++".

The data from the tests and interviews converge, supporting the inference that the use case form is least well understood by stakeholders despite their comfort with the format, the sequence diagram form only understood by those familiar with it but apparently well understood in these cases, and that ScenarioML is easily understood by stakeholders despite their inexperience with it. The interview data on ScenarioML is strikingly positive on this point.

7. Lessons Learned

Requirements specification formats vary in clarity for stakeholders.

This statement is so obvious that we did not think to make it until confronted by data that stakeholders do not consider all formats equally clear. This issue has received little consideration because non-developer stakeholders have not been the main target user group for requirements specification formats. In other words, it was assumed that stakeholders would find all formats equally *unclear* or confusing. This is not the case and we have cataloged ways in which formats can be unclear. For example, the numbering scheme for event sequences needs to be easy to follow and understand. Another problem can be found in graphical notations that obscure readability by visual overload and confusing notations that lack clear definitions.

Clarity for stakeholders is a significant quality.

Clarity of requirements specification formats for stakeholders is important because it affects how and how much they can participate in the requirements engineering process. In addition to elicitation, cooperative stakeholders are also needed for other activities including validation, verification and negotiation. Stakeholders and developers need to be able to find the information they need easily, understand that information, and know when they don't understand the information. Therefore, when selecting a format, requirements engineers should select one that affords the level of stakeholder participation that they desire.

Clarity for stakeholders can be operationalized.

Although communication is subjective, we have made clarity objective by finding ways to empirically measure stakeholder performance using different formats by testing their comprehension. By setting out goals, questions and metrics, we established an operational definition for clarity. These concepts can in turn be used as design objectives when improving requirements specification formats.

8. Conclusion

In this paper, we reported on an empirical study that revealed the importance of clarity in requirements. We compared three requirements specification formats, ScenarioML scenarios, use cases, and sequence diagrams. Our empirical study consisted of creating documents in each of the formats, collecting data on their creation, and evaluating the usability of these documents from the point of view of stakeholders. We compared the formats themselves, our experience working with the formats, and stakeholder preference and understanding of information in the formats. The data from the translation and user study indicated that ScenarioML was the format with the greatest clarity, followed by use cases, followed by sequence charts. Despite stakeholders' greater familiarity with use cases, they both preferred ScenarioML and found the format clearer. Our analytic results, stakeholders' subject opinions, and stakeholders' objective performance were a strong endorsement of the ScenarioML format.

While the research design that we used in this study provided us with novel insights into requirements specification formats, the results were qualitative in nature and not statistically generalizable. We plan to address this shortcoming in our next empirical study, which will be an experiment involving a much larger sample size. The working hypothesis is that material presented in a clearer format will be remembered better. We will be using a between-subjects combinatorial design where each subject is given a single format (ScenarioML, use cases, or sequence diagrams) and tested on their recognition and recall of details from the requirements. We expect to be able to perform tests of statistical significance on the quantitative performance measures from this planned study.

9. Acknowledgements

The authors wish to thank WebReach, Inc. for their generous cooperation in the study, and Suzanne K. Schaefer for helping us develop the study protocol. We also thank the anonymous reviewers for their careful and insightful comments.

References

[1] T. A. Alspaugh, B. Tomlinson, and E. Baumer. Using social agents to visualize software scenarios. In *ACM Symposium on Software Visualization (SoftVis'06)*, 2006.

[2] V. R. Basili, G. Caldiera, and H. D. Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*, pages 528–532. John Wiley and Sons, 1994.

[3] C. Ben Achour, C. Rolland, C. Souveyet, and N. A. Maiden. Guiding use case authoring: Results of an empirical study. In *Fourth IEEE International Symposium on Requirements Engineering (RE'99)*, pages 36–43, 1999.

[4] B. W. Boehm. *Software Engineering Economics*. 1981.

[5] F. P. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, 1987.

[6] A. Cockburn. *Writing Effective Use Cases*. 2000.

[7] J. W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. Sage Publications, 2003.

[8] A. M. Davis and A. S. Zweig. The missing piece of software development. *J. Syst. Softw.*, 53(3):205–206, 2000.

[9] B. G. Glaser and A. L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine, 1967.

[10] K. L. Heninger. Specifying software requirements for complex systems: New techniques and their application. *IEEE Transactions on Software Engineering*, 6(1):1–13, 1980.

[11] Mirth. `http://www.mirthproject.org/`.

[12] I. Sommerville. *Software Engineering*. 2006.

[13] Standish Group. The CHAOS report, 1994.

[14] K. S. Wasson. A case study in systematic improvement of langauge for requirements. In *14th Int. Conference on Requirements Engineering*, 2006.

[15] WebReach. `http://www.webreachinc.com`.

[16] J. Whittle. Specifying precise use cases. Technical Report ISE-TR-06-02, George Mason University, Feb. 2006.

[17] C. Williams, M. Kaplan, T. Klinger, and A. Paradkar. Toward engineered, useful use cases. *Object Technology*, 4(6):45–57, 2005.

[18] K. Winbladh, T. A. Alspaugh, D. J. Richardson, and R. Waltzman. Meeting the requirements and living up to expectations. Technical Report UCI-ISR-07-1, Institute for Software Research, University of California, Irvine, 2007.

[19] K. Winbladh, T. A. Alspaugh, H. Ziv, and D. J. Richardson. An automated approach for goal-driven, specification-based testing. In *International Conference on Automated Software Engineering (ASE'06)*, pages 289–292, 2006.