Exploiting Use Case Descriptions for Specification and Design - An Empirical Study

Karl Cox and Keith Phalp
Empirical Software Engineering Research Group
Bournemouth University, UK
(coxk; kphalp@bmth.ac.uk)

Abstract
Going beyond requirements, software engineers often employ use case descriptions to help them build initial design models of the proposed system. Despite Jacobson’s claim that “objects naturally fall out of use cases”, finding design-oriented classes in use case descriptions is not always straightforward. Indeed, the initial production of use cases may also prove problematic, and writing guidelines have been suggested as one way of improving the quality of use case descriptions.

This paper follows previous work (some of which has been presented at EASE), which compared writing guidelines in order to assess the comprehensibility of resulting descriptions. The work is distinct in that previous work has viewed the description wholly as a document for specification. This meant that descriptions were assessed in terms of issues such as clarity and consistent abstraction. However, for design, other issues come to the fore, notably what useful information can be gleaned from the description, to inform the architecture of the system. Therefore, this paper examines which guidelines produce descriptions that are most useful to inform design.

In brief, the results of this study suggest that those guidelines previously reported as most successful in the production of use case specifications (our own CP Rules), do not prove most useful for design. Hence, that designers would benefit from more detailed descriptions, especially concerning internal system events, and that this level of detail is most prevalent in descriptions produced using the CREWS Guidelines.

Finally, our findings seem to suggest that it may be beneficial to produce separate use case descriptions, employing different guidelines or structures, for requirements validation and for system specification and design.

Key words: use case description, specification, design, experiment

1 Introduction
This experiment continues a theme (presented at previous EASE conferences - Cox and Phalp (2000), Phalp and Cox (2002)) of examining the way in which guidelines for the production of use case descriptions have an impact upon the description. Existing studies have concentrated upon improvements to the use case description as a vehicle for specification. That is, upon using guidelines to improve the 'comprehensibility' of the resulting description. However, in considering such issues the intended audience for the description is, typically, the requirements engineer and customer. Hence, the clarity of the description is paramount, since a major issue is that of validation (Wieringa 2001).
However, this is only half of the picture, since use cases are often viewed as a vehicle to inform design (Jacobson et al. 1992, Rosenberg 1999, Insfran et al. 2002). That is, a designer may be presented with use case descriptions, and have to elicit information from them. Hence, this study seeks to address the extent to which use cases are effective in this undertaking. In particular, since previous studies showed that description guidelines were beneficial for specification, this study sought to examine the impact of these guidelines for the designer. In doing so, the CP Rules (Cox et al. 2001) and CREWS Guidelines (Achour et al. 1999) are again compared, since they represent the 'state of the art' in terms of writing guidelines.

Therefore, this paper presents an experiment that asks if use case descriptions produced with CP Rules can elicit more information than an equivalent CREWS Guidelines-based description. This is investigated on two levels. Firstly, by asking comprehension questions specific to a description and, secondly, by exploring what specification and design information can be elicited from a given description.

The following section (two) considers the design of the experiment, whilst section three reports experimental results. Section four considers threats to validity, section five discusses the findings, and finally section six suggests some conclusions.

2 Experimental Design

2.1 Subject Experience Pre-Experiment
This experiment used student subjects on an undergraduate course in Software Engineering. Before the experiment, the subjects had attended a module on software design. The subjects were reasonably versed in writing use case descriptions and drawing class diagrams. The subjects were asked, as a seminar task, to draw a class diagram from two use cases without any guidance except the typical noun / verb search approach (Rosenberg 1999). After this seminar, the subjects attended an hour-long lecture on finding classes from use cases and about identifying dependencies between description events and “hidden” classes in the description.

2.2 Course of the Experiment
The subjects were presented with a description to read and were required to answer a set of questions about that description. These were,

- specific to the subject of the use case (part one)
- general, regarding dependencies between events (part two)
- general eliciting of classes, their operations and attributes (part three)

Subjects asked the experimenters any questions they had before the writing tasks. They then had forty-five minutes to answer the questions. This was considered enough time to answer the questions. All experimental materials were collected at the end of the experiment.

2.3 Experimental Treatments
There were two experimental treatments:
1. Answer questions about an ATM use case description written following the CP Use Case Writing Rules.
2. Answer questions about an ATM use case description written following the CREWS Use Case Authoring Guidelines.

The two descriptions were the highest scoring descriptions from an earlier study (Cox et al. 2001). The descriptions were therefore not identical. However, each provided the best example of a description using each guideline set. Their use (as opposed to ‘manufactured’ descriptions) was necessary to produce meaningful findings and avoided introducing experimenter bias.

2.4 Experimental Subjects
There were 48 subjects, in two experimental groups of 24 subjects (Table 1).

<table>
<thead>
<tr>
<th>Experimental Group</th>
<th>Experimental Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>A (24)</td>
<td>CP Use Case</td>
</tr>
<tr>
<td>B (24)</td>
<td>CREWS Use Case</td>
</tr>
</tbody>
</table>

Table 1. Experimental Groups and Treatments

The subjects were assigned to experimental groups based upon their seminar groups. For instance, seminar groups A, C and E were assigned the CP treatment (experimental group A), and seminar groups B, D and F the CREWS treatment (experimental group B). The experimenters were unaware of the ability of the subjects prior to the experiment.

2.5 Experimental Hypotheses
There were three experimental hypotheses, based upon the question types,

\[ \text{H1}: \text{Use cases written with the CP Rules are more comprehensible than use cases written with the CREWS Guidelines.} \]

\[ \text{H2}: \text{CP Rules use cases allow for more identification of dependencies than CREWS use cases.} \]

\[ \text{H3}: \text{CP Rules use cases allow for more identification of classes than CREWS use cases.} \]

The null hypotheses state that there is no significant difference between the groups.

The hypotheses are one-tailed because since the CP ATM descriptions were more communicable than CREWS ATM descriptions from an earlier study (Cox et al. 2001), it is reasonable to expect that CP will perform better than CREWS here.

It is relatively straightforward to test hypothesis one. Subjects answer each comprehension question correctly, make an assumption or answer incorrectly. A correctly answered question carries two marks, an assumption (when considered a reasonable assumption) carries 1 mark and an incorrect answer (invalid assumption, incorrect or unanswered) carries 0 marks. An unpaired, 1-tailed t-test is used to test for any significant difference between the experimental groups.
Hypothesis two compares the number of identified dependencies by subjects with that of the authors’ ‘expert’ answers, as identified prior to the experiment. The quality of subject answers is poor, with a high number of unstated dependencies. This suggests that there might be an unequal distribution. The Mann-Whitney U test is therefore used instead of the t-test.

Hypothesis three is tested (unpaired, 1-tailed t-test) by comparing the classes identified by the experimental groups.

Any acceptable dependencies or classes not identified by the experimenters prior to the experiment are considered valid.

3 Experimental Results

3.1 Hypothesis One: Specific Questions
All subjects received the same set of questions. Table 2 shows no significant difference ($p = 0.94$) between the groups in answering the specific questions.

<table>
<thead>
<tr>
<th>Result</th>
<th>Group A (CP)</th>
<th>Group B (CREWS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>496</td>
<td>555</td>
</tr>
<tr>
<td>Mean total</td>
<td>20.67</td>
<td>23.13</td>
</tr>
<tr>
<td>Significance</td>
<td>p = 0.94</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Totals and means for groups A and B

The CREWS description contains some internal design events; the CP use case does not. Thus, group B answered more questions factually than made assumptions. The range of marks was very narrow for both groups. Figure 1 (left) shows that the majority of scores for group A are from 15 to 24 marks, whereas the majority for group B are from 20 to 30, showing that on average group B answered more correct questions or made more correct assumptions than group A.
3.1.1 Comments on the Answers Given
The questions mix abstractions, for instance, they ask what happens at the interface and what might happen inside the machine. One question asks where the new PIN is stored once it has been changed. The CP description does not state where this occurs, and seven subjects correctly answer that the description does not state this whereas the rest provided an assumption.

3.1.2 Hypothesis One Conclusions
Hypothesis 1 states that the CP Rules description will be more comprehensible than the CREWS description. This is determined by the number of correct answers given, assumptions made and incorrect responses. Statistically, there is no significant difference between the groups (p = 0.94). As such, the hypothesis is rejected.

Indeed, it is clear that the CREWS group scored higher marks than CP, probably because the CREWS description contained some internal design events. It would therefore be expected that group B (CREWS) answer the specific questions better than group A (CP). The CP description does hold to the ideal of only describing actor-focused interactions at the interface of the machine (Mattingly and Rao 1998). The CREWS description mixes abstractions. With hindsight, it is unsurprising, then, that group B scored better. However, since CP had fared better in terms of specification, it was plausible that the improved comprehensibility may also have provided dividends for design.

In terms of discovering internal design issues, these results suggest that it might be better to write descriptions that take the CREWS viewpoint, so that a scenario (or use case) can describe actions in the problem environment, at the interface and also inside the machine itself. That is, the more there is relevant detail, the easier it is to answer design questions. From the viewpoint of a customer, however, this is probably not that important, and might be seen as a poor approach to specification. This seems to suggest that descriptions should be written differently for different audiences.

3.2 Hypothesis Two: Identification of Dependencies Among Events
Hypothesis 2 states that the CP description will enable identification of more dependencies than the CREWS description. The Mann-Whitney U test is used because there are large amounts of unidentified dependencies suggesting there might not be a normal distribution.

Dependencies are defined as the necessary conditions that must be displayed by entities involved in the description for an event to occur and to complete. In essence, these are pre- and post-conditions of events in the description.

3.2.1 Group A (CP) Dependencies Identified
Group A identified one hundred and fifty-seven dependencies, an average of approximately 6.5 each (one subject, A17, did not find any, A19 scored 24 and A24 scored 16). None were identified for alternative and exceptional flows of events. Subjects possibly assumed that focus on the main flow of events was more important than any alternative sections. There were (at least) 30 unidentified pre- and post-conditions from the main flow (out of at least 68). (“At least” because some events have more than one pre- or post-condition.)
Overall, dependency identification was less than expected. This could be due to time constraints – there were many tasks to complete. It is also possible that subjects were unsure of what dependencies really meant and how one should identify them. This is questionable because the lecture prior to the experiment was focussed on this task and all subjects had attended that lecture.

3.2.2 Group B (CREWS) Dependencies Identified
Group B identified one hundred and forty-two dependencies, an average of approximately 6 per subject. Four subjects failed to identify any and subject B22 identified 32 dependencies. The nearest to this was B3 with 15.

3.2.3 Discussion on Dependency Discovery
A Mann-Whitney U test (where $p = 0.12$ and ties are omitted) shows there is no significant difference between the groups. Figure 2 describes the frequency of identified dependencies, showing that just over half of the dependencies for the CP use case description are not identified. That is, there are 40 events in the description, each with (at least) one pre- and post-condition, making 80 in total. 42 of these were not identified. As examples of those identified (figure 7-2 right), there are 11 dependencies that are identified once; 4 that are identified twice etc.

![Figure 2. Pie chart for group A (left) for frequency of identified dependencies (right)](image)

![Figure 3. Pie chart for group B (left) for frequency of identified dependencies (right)](image)
Figure 3 (left) shows that group B’s subjects failed to identify approximately two thirds of dependencies but that overall there is approximately the same number of dependencies identified (48, figure 3 right) compared against group A (38, figure 2 right).

3.2.4 Hypothesis Two Conclusions
Hypothesis 2 states that more dependencies can be identified from a CP use case description than a CREWS description. Although group A (CP) finds more dependencies than group B (CREWS), the difference is not significant (p = 0.12), with group A averaging 6.5 per subject and group B, 6.

The identification of dependencies is not a particularly easy task, and it is rather time-consuming, indeed there is the threat of “analysis paralysis” (Booch 1994, p.6). This could be proposed as a reason why the experimental subjects did poorly. However, determining how the system is supposed to respond to events at the interface is a significant aspect of design, and examining dependencies is one way to facilitate this process. Examination of dependencies acts as a validation mechanism for the use case description. For example, the software engineer has to be confident that the event under scrutiny is correct to enable movement to the next event.

Overall, we found no significant differences between the group. However, it is perhaps worth noting that both performed poorly. Since dependencies are important to proper validation (Sutcliffe 1998), this suggests that further work in helping to improve the identification of dependencies within use case descriptions is vital. Indeed, this support for identification of dependencies within use cases is now a topic of further work within the group.

3.3 Hypothesis Three: Identification of Classes

3.3.1 Classes Found by Group A (CP)

Table 3 shows the total number of correct and erroneous classes identified by groups A and B that match the experimenters’ answer.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Group A (CP)</th>
<th>Group B (CREWS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctly identified classes</td>
<td>41</td>
<td>77</td>
</tr>
<tr>
<td>Erroneous classes</td>
<td>31</td>
<td>53</td>
</tr>
<tr>
<td>Correctly identified operations</td>
<td>76</td>
<td>70</td>
</tr>
<tr>
<td>Misplaced operations</td>
<td>160</td>
<td>137</td>
</tr>
<tr>
<td>Correctly identified attributes</td>
<td>30</td>
<td>42</td>
</tr>
<tr>
<td>Misplaced attributes</td>
<td>66</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 3. Groups A and B classes, operations, attributes identified

‘Correctly identified operations’ and ‘correctly identified attributes’ refer to those operations and attributes that are part of a correctly identified class. ‘Misplaced’ operations and attributes are those that are correct to the diagram but incorrectly located within the wrong class.
The results are not particularly encouraging for group A since the average correct class identification is less than two. The two most frequently identified classes were Customer (sometimes User) and Card (9 subjects identified these). In terms of this level of abstraction, it is questionable whether the Customer and Card should be there at all. Indeed, this might be considered as “naïve real-world modelling” (Isoda 2001, p.155). Ideally, these classes ought to be viewed as actors since at this level of abstraction the classes inside the machine that refer to real-world entities should only hold information about those real-world entities and not be a direct representation of them. This suggests two possibilities. First, group A’s subjects were not particularly good at object-oriented analysis. However, time constraints reduced the think time that the subjects had for class identification compared to an industrial context. Second, the identification of classes from use case descriptions might not be that obvious. The description might need internal design events to help identify classes. Though we argue internal design should be kept out of descriptions for requirements purposes, their inclusion might be beneficial to design. Once again, this suggests a need to produce separate use cases for customer validation and for design.

3.3.2 Classes Found by Group B (CREWS)
Table 3 shows the total number of classes identified by group B. Group B correctly identified 77 classes, an average of 3 per subject. 16 subjects identified the two problem domain (analysis) classes Customer and Card. 21 subjects incorrectly identified the ATM as a class. Like group A, there were a large number of misplaced operations (137). This is perhaps unsurprising since a use case description is primarily about events, which can be roughly translated into operations. 9 subjects identified a further acceptable class Keypad (compared to 0 from group A), which had not been in the 'expert' solution.

3.3.3 Comparing Groups
Group B identified almost double the number of correct classes compared to group A (77 to 41). When an unpaired t-test is applied, hypothesis three is rejected (p = 0.99). This suggests that group B identified significantly more classes than group A and reflects, to a degree, on the abstraction represented in the descriptions presented to both groups. The difference between the groups is also shown by the identification of other classes, not least that of the Customer and Card classes which group B identified almost twice as much as group A (16 subjects in group B to 9 in A).

Figure 4. Histograms for groups A (left) and B (right) of number of classes identified.
Figure 4 describes histograms for groups A and B. The histogram for group A (left) shows that all but one of its subjects identified 3 classes or less and 5 subjects identified none. It is disappointing that these subjects identified no classes – it might be that they were not certain what a class might be in the context of the description. In comparison, all of group B’s subjects (right) identified at least a single class. 9 subjects identified four or more classes.

The large numbers of operations and attributes identified overall by both groups (table 3) suggests that the subjects had enough time to answer the questions but that they were not particularly good at doing object-oriented analysis. Perhaps the student experiment was not necessarily as successful as hoped and that it might not reflect the abilities of experienced practitioners.

3.3.4 Hypothesis Three Conclusions
Hypothesis 3 states that the CP description will lead to more discovered classes than the CREWS description. This is shown not to be the case. In fact, group B (CREWS) identified almost twice as many classes as group A (p = 0.99). This can be seen as a significant difference in favour of group B. The difference does appear to be aided by the level of abstraction described in the use case descriptions. The hypothesis is rejected.

4 Threats to Validity

4.1 Statistical Power
The statistical power could not be assessed because it was very difficult to judge what the effect size might be. This would determine the number of required subjects. Miller et al. (1997) suggest that to determine alpha at 0.05, there should be 15 subjects per group. Since there are 24 per group, this is deemed as sufficient to set alpha at 0.05.

4.2 History
Not all subjects participated in the experiment at the same time. The experiment occurred in separate sessions over a week, due to time-tabling restrictions. There is a risk that subjects would pass on any knowledge to subjects yet to take part (causing diffusion of treatments). The only reasonable way to control this was by retrieving all experimental material at the end of each experiment session.

4.3 Maturation and Mortality
The experiment lasted an hour (the writing part 45 minutes). Concerns over boredom or over-enthusiasm were not considered significant. No subjects dropped out of the experiment. It is probably the case that subjects were under time pressure. After answering the specific questions (section 3.1) and examining dependencies (section 3.2) it might be that the subjects were over-tasked. However, there are a large number of classes and operations identified overall (section 3.3). As such, subjects did appear to have time to answer the questions.

4.4 Inadequate Preoperational Explication of Constructs (Wohlin et al. 2000)
All hypotheses were significance tested. Both experimenters checked all measures and agreed on the 'correct' answers and on those responses that were deemed correct but initially missed by the experimenters. It is unlikely, therefore, that the measures (correct answers to H1, dependencies in H2 and classes for H3) are incorrectly designed.
4.5 External Validity
The results of the experiment cannot be easily generalised to every software company employing use case descriptions, because software practitioners might be expected to be more experienced. However, the results might be generalisable of undergraduate software engineering students who have taken a course in software design for the first time.

4.5.1 Nature of the Problem (Host et al. 2000)
The ATM example is used because the CP ATM description scored the best marks in an earlier experiment (Cox et al. 2001) and for comparative purposes it was necessary to present the other experimental group with the best CREWS ATM description. The point of using these descriptions instead of fabricating one’s own was simply to avoid introducing bias into the experiment. The ATM is still a popular choice of example for use case descriptions (e.g. Kosters et al. 2001, Rolland 2002) and use of an ATM was expected to be familiar to the subjects; it is thus considered valid to use here.

4.5.2 Setting (Robson 1993)
Engineers in software houses would probably interrogate descriptions and would probably have other distractions so that a complete interrogation might take place over time. It is also probable that a practitioner would validate their answers with a colleague. Therefore, the classroom at a university is not a typical setting, though it is for conducting experiments.

5 Discussion
The results suggest that for more correct class identifications, descriptions should contain lower levels of abstraction; this can be considered a description from the designer’s viewpoint. The larger number of classes identified by group B points to this being the case. However, the OMG (2001) states, “use cases … indirectly state the requirements the specified entity poses on its users; that is, how they should interact so the entity will be able to perform its services.” (p.2-141, authors’ italics). The OMG, though, avoids being specific about mixing abstractions in one use case description. It appears to be the case that when a description is used by a designer, it is be better to contain internal design events since this is the kind of information the designer would look for. Indeed, it has been shown that designers think and move through different levels of abstraction when considering a design problem (Guindon et al. 1987) and thus if use case descriptions contain mixed abstraction levels, then this might aid the designer in constructing a design.

Not all classes were identified and there were a large number of incorrectly placed operations and attributes. It is clear, though, that the notion of interrogating descriptions in this way does yield a lot of design-oriented information. It also makes one think more about the system and how it relates to the problem environment. Even if descriptions are going to be at more than one level of abstraction, additional questions that enable better interrogation of aspects of descriptions are worth exploring further.

5.1 Further Work
The questions presented to experimental subjects need to be expanded upon to take into consideration further aspects of today’s software systems. To have more coverage of issues of design, for instance, the questions ought to consider connections to other software systems. In addition, this study suggests that separate guideline sets may prove fruitful for different audiences (e.g., requirements engineer versus designer). Hence, the
identification of which parts of guidelines provide details at differing levels of abstraction needs further investigation.

6 Conclusions
This paper described an experiment that compared the comprehension of two use case descriptions from the perspective of using such descriptions to inform design. The work followed from earlier studies, which had investigated the impact of guidelines on the production of use case descriptions from a requirements validation perspective. Previous studies had demonstrated, firstly, that use case descriptions produced using guidelines were more comprehensible descriptions than those produced without, and secondly that the 'smaller' CP Rules performed at least as well (or better) than the larger set of CREWS Guidelines.

Hence, these two leading guideline sets were again the focus of the investigation. Three hypotheses stated that the CP use case would, first, be more comprehensible than the CREWS use case; second, that it would find more dependencies than CREWS; and third, that it would find more classes than CREWS. All hypotheses were rejected in favour of the null. It was found that CREWS did better than CP in all hypotheses except for the identification of dependencies.

On investigation of the results, it appeared that the low-level detail found in the (mixed abstraction level) descriptions produced with CREWS Guidelines aided the elicitation of design details. With hindsight this is perhaps not so surprising. However, this result raises a number of issues. In particular it confirms that, as with traditional approaches to requirements and specification, the form of use case most beneficial for requirements gathering and validation is different to that most beneficial for the move towards design. This suggests that different use case forms and different guidelines may need to be developed which specifically target the exact phase of the requirements and specification process.

References


