Quality and Understandability of Use Case Models

Bente Anda[1], Dag Sjøberg[1,2] and Magne Jørgensen[1]

[1]Department of Informatics
University of Oslo
P.O. Box 1080 Blindern
0316 Oslo
NORWAY
{bentea,dagsj,magnej}@ifi.uio.no

[2]Simula Research Laboratory
P.O. Box 1080 Blindern
N-0316 Oslo
NORWAY

Abstract. Use case models are used in object-oriented analysis for capturing and describing the functional requirements of a system. Use case models are also used in communication between stakeholders in development projects. It is therefore important that the use case models are constructed in such a way that they support the development process and promote a good understanding of the requirements among the stakeholders. Despite this, there are few guidelines on how to construct use case models.

This paper describes an explorative study where three different sets of guidelines were used for constructing and documenting use case models. An experiment with 139 undergraduate students divided into 31 groups was conducted. Each group used one out of the three sets of guidelines when constructing a use case model from an informal requirements specification. After completing the use case model, each student answered a questionnaire.

The results of the experiment indicate that guidelines based on templates support the construction of use case models that are easier to understand for the readers, than guidelines without specific details on how to document each use case. The guidelines based on templates were also considered as the most useful when constructing use cases. In addition to better understandability, our experiment indicates that the guidelines based on templates result in better use case models regarding also other quality attributes. Our results further indicate that it may be beneficial to combine the template guidelines with another set of guidelines that focus on the documentation of the flow of events of each use case.

Keywords. Object-oriented analysis, Requirements specification, Use Cases, UML, Understandability, Experiment

1 Introduction

The concept of use case modelling was introduced by Jacobson in 1992 [1]. He also introduced a use case driven approach to software development. Since then, use case modelling has become a popular and widely used technique for capturing and describing the functional requirements of a software system. It is also used as a technique for bridging the gap between descriptions that are meaningful to software users and descriptions that contain sufficient details for modelling and constructing a software system. A use case model has two parts, the use case diagram and the use case descriptions. The diagram provides an overview of actors and use cases, and their interactions. The use cases' text details the requirements. An actor represents a role that the user can play with regard to the system, and a use case represents an interaction between an actor and the system.

In a use case driven software development process, the use case model is recommended as a primary artefact and is, for example, input to design and a basis for planning, estimation, testing and documentation. In addition, a use case model will often be part of a contract between the development organization and the customer regarding the functional requirements of the system to be developed. Therefore, the quality of the use case model may have a large impact on the quality of the resulting software system. Nevertheless, there is no commonly agreed theory on how to construct use cases, and there are different opinions about what constitutes quality in use case models.

UML adopts use cases but offers little advice on how to apply them. The support for use case modelling in most UML CASE-tools is also limited. For example, they do not support traceability from use cases to other diagrams even though it is recommended in use case driven development.

Quality attributes of use case models and advice on how to construct them have been proposed in the literature [1-14]. These recommendations are mostly based on extensive experience from software projects. However, to our knowledge only the guidelines for use case authoring developed in the CREWS project [14] have been subject to empirical evaluation.

Use case models are frequently claimed to be easy to understand for the stakeholders involved in a development project [1-3], and a good understanding of the use case model is important if the use case model is to contribute successfully to the development project. In our opinion, understandability is therefore an important quality in use case models. Moreover, the large number of, sometimes contradictory, guidelines for use case modelling to choose from motivate the overall objective of this study: To empirically investigate the effect of different guidelines on the quality, in particular understandability, of a use case model.

We conducted an experiment with 139 students divided into 31 groups. The groups were organized in pairs; one group was the customer for a system to be developed, while the other group was the development team. The development teams used one out of three different sets of guidelines in the construction of a use case model for the system. The use case models were evaluated by the authors of this paper according to a set of quality attributes. To evaluate understandability, the students answered questions about the functionality in the use case models. The students were also asked about how useful they found the guidelines.

The remainder of this paper is organized as follows. Section 2 gives an overview of different guidelines and recommendations on use case modelling together with suggestions for properties of quality. Section 3 describes our experiment in details. Our results are presented in Section 4 and discussed in Section 5. Section 6 discusses threats to the validity of the results. Ethical considerations relevant to this experiment are discussed in Section 7. Section 8 concludes and describes future work.

2 Use Case Modelling

Use cases can be described in many ways ranging from an informal, unstructured style to a more formal style approaching pseudocode [16]. Different organizations construct and apply use case models differently [13]. Independently of the format and notation, deciding a suitable level of detail is a challenge. A too fine granularity makes the use case model difficult to grasp, while a too coarse granularity hides the complexity [2].

Cockburn [5] recommends that the format should be chosen for each project, and that the choice should be driven by both characteristics of the development team and the main purpose of the use case model. The level of experience in the team, both regarding application domain and regarding use case modelling, is a relevant factor because little experience may require more support from the use case format. Another relevant factor is the cohesiveness of the team since a team working closely together can write more casual use cases than a larger, perhaps geographically dispersed team.

The future use of a use case model in a development project should also be an important issue when determining the appropriate format. An example is applying use case models in estimating effort for software development projects [6,17,18]. We have conducted a case study that indicates that the format of the use case model impacts the estimates [18].

Different formats may also support other activities differently, for example, the ability to identify classes during design or the ability to validate and verify an architectural decision.

There are, in our opinion, challenges at three levels when constructing a use case model. All three levels may be improved by appropriate guidelines. In our experiment, we test one guideline at each of the three levels, which are respectively described in Sections 2.1, 2.2 and 2.3. Section 2.4 describes some quality attributes of use case models.

2.1 Minor Guidelines

Actors are identified by considering who will receive information from a system and who will provide it with information. Use cases are identified by asking what are the main tasks of each actor. The first set of guidelines used in our experiment, called Minor guidelines (alternative 1 in Appendix A), are based on those found in [1,6]. They describe how to identify actors and use cases, but give little direction on how to construct them.

We included this alternative in our experiment primarily because we wanted to investigate how the students documented their use case models when they were not given specific guidelines on how to do it. Another purpose was to study how a more elaborate description of how to identify actors and use cases affected the resulting use case models.

2.2 Template Guidelines

A template is often recommended for documenting use cases because the predefined structure of a template forces the developer to identify and include important elements in each use case [2]. Examples of templates can be found in [2,5,6,10,11,12]; their most typical content is shown in Table 1. Our second set of guidelines is thus based on templates. The Template guidelines (alternative 2 in Appendix A) include a template for describing actors and a template for describing for use cases. These templates are based on the templates used in [2,5,6].

Table 1. Content of templates

Property	**[2]**	**[5]**	**[6]**	**[10]**	**[11]**	**[12]**
Title	x	x	x	x	x	x
Actor(s)		x	x	x	x	x
Trigger	x	x				
Scope		x				
Summary	x		x	x	x	x
Preconditions	x	x	x	x	x	x
Basic flow of events	x	x	x	x	x	x
Extension points	x	x	x	x	x	
Alternate courses	x	x	x	x	x	x
Post-condition	x	x	x		x	x

2.3 Style Guidelines

There are different recommendations on how to structure the description of the flow of events. In [1] and [8], narrative text with alternatives and extensions is recommended. Cockburn [5] recommends that users should warm up with narratives, but that the final use cases should follow a predefined template, as narrative use cases are often ambiguous and lack structure.

The guidelines described in [14] focus on the individual events in the flow of events. The aim is to give each description of events a standard structure and thereby make them easier to read. There seems to be an agreement on the following advice on how to describe the flow of events and each single event:

- Each event should be numbered in order to show how the process moves forward.
- Each event should clearly show which agent is active.

- Each event should be described with a present tense verb phrase in active voice.
- The user's vocabulary should be used.
- Repetition in the flow of events should be handled with a *while*, *for* or *repeat..until* construction.

However, there is some disagreement on the following:

- Whether the flow of events should be described in one section with narrative text, as a numbered list of events or as a list of event-response pairs.
- The handling of alternatives. In [6] and [14] an *if..then* construction is recommended, while such a construct is warned against in [2].
- The use of pseudocode in the description of each event. The structure recommended by Ben Achour *et al*. [14] strongly resembles pseudocode, while Kulak and Guiney [2] claim that pseudocode is too different from the user's language.

We have used a modified version of the guidelines described in [14], which we call Style guidelines[1] (alternative 3 in Appendix A) as the third set of guidelines in our experiment. We have modified the guidelines slightly based on the results from the experiment described in [15]. That experiment indicated that some of the original guidelines were difficult to use since they were implemented by few of the participants, and some were superfluous since all the participants, including those who did not receive the guidelines, implemented them. We decided to use these guidelines because they had been subject to evaluation in two former controlled experiments and could thus be evaluated in a different context in our experiment.

2.4 Recommendations for Use Case Models

It is believed that the quality of use case models has an impact on the quality of other documents subsequently produced in the development process. Hence, it seems sensible to investigate which properties of a use case model contribute to its quality. Many recommendations exist, for example, the following found in [2,8,12-14]:

- The use case should be easy to read.
- The descriptions should not include any assumptions of design or implementation.
- The descriptions should not include interface details.
- Events that are not related to the overall goal of the use case should not be described.
- The action descriptions should be complete.
- The flow structure should be correct and unambiguous.
- The terminology should be consistent.

We have used these recommendations as a basis for the attributes against which the use case models in our experiment are evaluated.

---

[1] The original proposers of the guidelines called them guidelines for style and content. For simplicity reasons we will denote them just Style guidelines.

3 The Experiment

This section describes the participants in our experiment, how they were trained, the detailed procedure of this experiment and the marking scheme. Finally, the hypotheses are presented.

3.1 Experiment Participants

The experiment was conducted as part of a compulsory project in an undergraduate course in software engineering with 139 students divided into 31 groups. We conducted a survey to identify the students' background.

The students were mostly aged between 20 and 30. Out of the 139 students, 43 had some experience from professional software development, while 9 had a lot of experience. There were 20 students who had professional experience with requirements engineering, 4 of them had done use case modelling. None of the students were familiar with any of the guidelines used in this experiment.

The students with experience in either project management or the languages used in the course (UML and Java) were *evenly* distributed among the groups. The other students were *randomly* assigned to the groups. Making the groups as similar as possible was important to study the effects of the guidelines.

The groups were organized in pairs with one group having the role of customer for the system to be developed, while the other group had the role of development team. Each group participated in two pairs, in one pair as customer for either system A or B (see appendix B) and in the other pair as development team for the other system. There was one exception; one group was both customer and developer for system A because of the odd number of groups.

3.2 Training

The software engineering course consisted of lectures and seminars. The students were divided into six seminar groups led by graduate students. Requirements engineering was taught in a one-hour lecture; basic use case modelling was taught in another one-hour lecture and in a one-hour seminar. The students were taught the concepts of actors and use cases, and they were given two examples (the same two for all groups) on how to describe use cases and their flow of events. In another one-hour seminar, the students were taught the guidelines and examples of how to use them.

3.3 Procedure of Experiment

The first activity in the student project consisted in creating an informal requirements specification based on the description of the system for which they were customers. This requirements specification was then handed on to their development team. The development teams made a detailed requirements specification including a use case model. The groups had two weeks available for this task, and they had the opportunity

to ask lecturers, seminar leaders and fellow students for help. It was recommended that they should talk to their customer in order to clarify their requirements.

When the requirements specification was completed, it was made available to the customer group. Some groups received use case models written using the same guidelines as they had used themselves, while others received use case models written using different guidelines.

The students then individually answered a questionnaire (shown in appendix C) with questions about the functionality in the use case model they had made themselves and about the functionality in the use case model they received from their development team. The questions were constructed based on the original system description. The questionnaire also included a question about how useful they found the guidelines, which was answered by ticking one of the following alternatives: *Very useful*, *quite useful*, *quite useless* or *very useless*. There were also questions about how much time they had spent working on both the informal and the formal specification, on communication with their development team and on reading through the use case model from the development team. The students had previously been asked to record effort so it should be possible for them to recapture the time used. It was compulsory to answer the questionnaire, and it was done at times normally scheduled for seminars. The students were given directions on how to answer the questions, and one of the authors was available for questions when they answered the questionnaires.

Answers to questions about functionality were used in another experiment to evaluate differences in understandability between requirements expressed in natural language and requirements expressed in an activity diagram [19]. In that experiment the participants were asked to tick the correct answer to a question from a list of possible an answers. In our experiment it was infeasible to give the participant a predefined list of alternative answers because we did not know the actual use cases beforehand. Instead the participants answered using free text. The questionnaire included two blank lines after each question to give the participants an indication of the length expected for their answers.

When answering the questionnaire, all the students had available a copy of their own requirements specification and the one made by their development team. On average they spent 40 minutes on these questions.

3.4 Marking Scheme

The resulting use case models were evaluated according to a list of different properties inspired both by the recommendations on how to write use cases described in Section 2 and by the marking schemes for flow of events suggested in [14,15]. Ben Achour *et al.* [14] used the following marking scheme in their experiment for evaluating the CREWS guidelines:

- Completeness for each event, determined by counting the amount of events that included agents, objects, communication sources and destination
- Completeness of the whole flow of events, determined by counting irrelevant action descriptions compared with a use case made by an expert on the application domain (a low count gave a high score)

- Number of correctly placed variations
- Number of homonyms and synonyms (a low count gave a high score)

In our experiment there were very few missing elements in the descriptions, so we decided not to include this aspect. Since our use case models were constructed from different informal specifications, we did not have the opportunity to compare the individual use cases with ideal solutions, but in our opinion there were very few irrelevant descriptions of events. We did, however, find missing or unrealistic descriptions, so the realism of the description of the flow of events was also considered in our evaluation. There were significant differences in how the students handled variations in the use cases. We have therefore counted both the number of variations for each use case model and the number of correct placements in the flow of events. The students used inconsistent terminology in the use case models. We therefore also included this aspect in our evaluation.

Cox & Phalp [15] describe a replication of the experiment conducted by Ben Achour *et al*. [14]. They found the original marking scheme difficult to use because it was too detailed to give a good overall assessment of the individual use cases. In addition to the marking scheme above, they therefore also evaluated the use cases more subjectively according to:

- Plausibility – the realism of the use case
- Readability – the flow of the use case
- Consistent structure – consistent terminology and use of present simple tense
- Alternative flow – consideration of variations

We found all these aspects relevant and therefore included them in our evaluation. In the two experiments described above, only single use cases were evaluated. In our experiment, complete use case models were constructed, each containing several use cases. Since we considered also aspects of the overall model, and because of the large number of use cases in our experiment, we used a slightly different marking scheme, which is based on a more overall evaluation of the use cases:

- Single diagram – the use case model should include one single diagram showing all the actors and use cases.
- Actors – the correct actors were identified. Correctness was determined relative to the informal requirements specification.
- Use cases – the correct use cases were identified. Correctness was determined relative to the informal requirements specification as above.
- Content – the description of each use case contained the information required by all the sets of guidelines: actor, assumptions that must be valid before the use case starts, flow of events, variations and post-conditions.
- Level of detail – the descriptions of each event were at an appropriate level of detail. There should be no unnecessary details about user interface or internal design. Each event should be atomic, that is, sentences with more than two clauses should be avoided.
- Realism – the flow of events was realistic, that is, the events follow a logical and complete sequence, and it is clearly stated where variations can occur.
- Consistency – the use of terminology was consistent.

Table 2. Properties supported by guidelines

Type of guideline	Single diagram	Actors	Use cases	Con-tent	Level of detail	Rea-lism	Con-sistency
Minor guidelines	x	x	x				
Template guidelines	x			x			
Style guidelines	x				x	x	x

Table 3. Marking scheme

Property	Mark	Comment
Single diagram	0-1	1 = correct, 0 = wrong
Actors	0-3	3 = all correct, 0 = all wrong
Use cases	0-3	3 = all correct, 0 = all wrong
Content	0-3	3 = all correct, 0 = all wrong
Level of detail	0-3	3 = all correct, 0 = all wrong
Realism	0-3	3 = all correct, 0 = all wrong
Consistency	0-2	2 = all correct, 0 = all wrong

The guidelines gave different support for these properties. Table 2 shows which guidelines support which properties. Each use case model was given a mark for each of these properties based on an overall assessment according to the marking scheme in Table 3.

The size of the use case models is also measured. Size is measured as a vector:

< *Number of actors,*
number of use cases,
median number of actions in the flow of events,
median number of variations >

We believe that the number of identified actors and use cases, together with the number of events and variations, indicate quality of the guidelines – the higher number, the better quality.

3.5 Hypotheses

This section presents the hypotheses tested in this experiment. Our first hypothesis ($H1_1$) is that different guidelines for constructing use case models have different effect on how well the use case models are understood by their readers.

$H1_0$: There *is no* difference in understanding when reading use case models constructed with different guidelines.

$H1_1$: There *is* a difference in understanding when reading use case models constructed with different guidelines.

Our second hypothesis ($H2_1$) is that the different guidelines have different effect on the understanding of the requirements from the point of view of those who use the guidelines to construct use case models.

$H2_0$: There *is no* difference in the understanding of the requirements when using different guidelines in the construction of use case models.

$H2_1$: There *is* a difference in the understanding of the requirements when using different guidelines in the construction of use case models.

Our third hypothesis ($H3_1$) is that the different guidelines are of different usefulness to those who construct the use case models.

$H3_0$: There *is no* difference in the usefulness of the different guidelines when constructing use case models.

$H3_1$: There *is* a difference in the usefulness of the different guidelines when constructing use case models.

4 Results

This section describes the results from the testing of the hypotheses. We used the non-parametric Kruskal-Wallis test since our data sets were not normally distributed. Results of the effect of the guidelines on other quality attributes are also presented.

4.1 Assessment of Understandability

After reading the completed questionnaires, we found that many of the questions about the functionality were irrelevant for the use case models describing system A. The reason was that many of them included other functionality than we had expected from the original description (Appendix B). We therefore consider only system B in the analysis of the answers about functionality.

The answers to the questions about functionality were compared with the functionality actually described in the use case models in order to determine their correctness. Each answer was given a mark of 0 (wrong answer or no answer), 1 (correct answer to a simple question or partially correct answer to a complicated question) or 2 (correct answer to a difficult question). Questions 1, 2 and 7 for system B (Appendix C) were classified as simple. The answers for system B could obtain a maximum of 13 points.

An example of a partially correct answer is when question B.2, “How is the roster made and updated, and who is responsible for it,” has the answer “unit nursing officer.” This answers the second part of the question but not the first part. The guidelines were compared regarding the score on correct answers for each individual customer and individual developer.

Table 4 shows the number of students who read the use case models for system B, distributed by the guidelines used in the construction of these models; similarly for the use case model construction. This table also shows the minimum, median, maximum and standard deviation of the scores on correct answers given by the readers and constructors, respectively.

Table 4 Descriptive statistics on the data used in the assessment of understandability

Type of guideline	Read-ing	Scores on reading				Const-ructing	Scores on constructing			
		Min	Med-ian	Max	Std		Min	Med-ian	Max	Std
Minor guidelines	14	2	6	11	2,6	13	5	8	12	2,4
Template guidelines	26	5	9	12	2,1	25	4	9	12	2,5
Style guidelines	27	1	8,5	13	2,9	28	2	9	13	2,7
Total	68					66				

Reading Use Case Models – Hypothesis H1. There was a significant difference in the score on correct answers between the customers who had read use case models constructed by developers who had used either the Template or Style guidelines compared with those who had used the Minor guidelines (Figure 1). There was no significant difference between the Template and Style guidelines, although the Template guidelines were slightly better.

The level of significance (alpha-level) chosen for this test was 0.05. The p-value of 0,021 obtained from the test signifies that we can reject $H1_0$ and that we can assume with a 95% probability that there is a difference between the guidelines.

```
Guidelines    N    Median    Ave Rank         Z
Minor        14     6,000        22,2     -2,61
Template     26     9,000        40,1      1,85
Style        28     8,500        35,4      0,32
Overall    68                   34,5

H = 7,58  DF = 2  P = 0,023
H = 7,70  DF = 2  P = 0,021 (adjusted for ties)
```

Fig. 1. Kruskal-Wallis test on correct answers for customers who had read use case models for system B

Half of the customers read use case models constructed using the same guidelines as they had used themselves in the role of developers. The other half read use case models constructed with guidelines with which they were unfamiliar. We investigated whether there was a difference in understanding related to whether the guidelines were familiar.

When the guidelines were familiar, there was a significant difference in favour of the Template guidelines (Figure 2). However, the number of subjects for the Style guidelines was much higher than for Minor or Template guidelines (24 vs. 5 and 6). This may explain why the Style guidelines did worse than when all the customers for System B were considered.

Guidelines	N	Median	Ave Rank	Z
Minor	5	9,000	15,3	-0,64
Template	6	11,000	28,8	2,84
Style	24	8,000	15,9	-1,83
Overall	35		18,0	

H = 8,11 DF = 2 P = 0,017
H = 8,49 DF = 2 **P = 0,014** (adjusted for ties)

Fig. 2. Kruskal-Wallis test on correct answers for customers who had read use case models for system B constructed with the same guidelines as they had used themselves

When the guidelines were unfamiliar, the Style guidelines apparently did well (Figure 3). However, this sample is very small (4), which means that we should not draw any conclusions on the effect of the Style guidelines from this test. We repeated the test with only the Minor and Template guidelines, and found a significant difference (p = 0,004) in favour of the Template guidelines.

Guidelines	N	Median	Ave Rank	Z
Minor	9	6,000	8,2	-3,19
Template	20	8,000	18,7	1,25
Style	4	11,000	28,3	2,48
Overall	33		17,0	

H = 13,45 DF = 2 P = 0,001
H = 13,73 DF = 2 **P = 0,001** (adjusted for ties)

* NOTE * One or more small samples

Fig. 3. Kruskal-Wallis test on correct answers for customers who had read use case models for system B constructed with different guidelines than they had used themselves

Understanding Requirements – Hypothesis H2. There were no significant differences between the guidelines when we compared the scores of the developers on the questions about functionality in the use case models they had constructed themselves (Figure 4). This indicates that the understanding of the requirements among those who had developed a use case model, depends primarily on other factors than the guidelines used when they constructed the use case model. The p-value of 0,835 obtained from the test indicates that we cannot reject $H2_0$.

Guidelines	N	Median	Ave Rank	Z
Minor	13	8,000	31,0	-0,53
Template	25	9,000	33,4	-0.04
Style	28	9,000	34,8	0,47
Overall	66		18,0	

H = 0,35 DF = 2 P = 0,838
H = 0,36 DF = 2 **P = 0,835** (adjusted for ties)

Fig. 4. Kruskal-Wallis test on correct answers for developers who had constructed use case models for System B

4.2 Assessment of Usefulness – Hypothesis H3

The questionnaire given to the students also included a question about how useful they found the guidelines. This question was answered by ticking one of four alternatives and each alternative was given a mark: *very useful* = 3, *quite useful* =2, *quite useless* =1 and *very useless* = 0. This question was equally relevant to both system A and B. All the questionnaires were therefore included in this analysis. This gave a total of 138 subjects (one of the students did not answered this question).

The p-value of 0,113 obtained from this test gives a weak indication that the Template guidelines were found most useful. We repeated the test with only the Minor and Template guidelines, and found a significant difference in favour of the Template guidelines (p=0,039). This indicates that we can reject $H3_0$.

Guidelines	N	Median	Ave Rank	Z
Minor	33	2,000	61,7	-1,29
Template	44	2,000	77,8	1,68
Style	61	2,000	67,7	-0,46
Overall	138		69,5	

H = 3,31 DF = 2 P = 0,191
H = 4,37 DF = 2 **P = 0,113** (adjusted for ties)

Fig. 5. Kruskal-Wallis test for developers on how useful they found the guidelines they had used

4.3 Assessment of Quality

The use case models constructed by the students in our experiment were evaluated according to the marking scheme described in Table 3, Section 3.4. Table 5 shows that use case models constructed using the Template guidelines obtained the highest overall score on the quality attributes. The developers using the Minor guidelines were best at understanding that they should make one single diagram, and they did better than those using the Style guidelines on identifying the correct actors and use cases. The use case models constructed using the Style guidelines obtained the highest score on consistency, that is, consistent use of terminology, and they did quite well on level of detail and realism.

Table 5. Average score on the properties of quality

Type of guideline	Single diagram	Actors	Use cases	Content	Level of det.	Realism	Consistency	Sum
Minor guidelines	Best (0,9)	Mid (2,3)	Mid (2,1)	Worst (1,1)	Worst (1,7)	Worst (1,7)	Mid (1,7)	Worst (11,3)
Template guidelines	Mid (0,8)	Best (2,6)	Best 2,5	Best (2,5)	Best (2,2)	Best (2,4)	Mid (1,7)	Best (14,6)
Style guidelines	Worst (0,6)	Worst (2,2)	Worst (1,8)	Mid (1,7)	Mid (1,9)	Mid (2,0)	Best (1,8)	Mid (12,0)

Table 6 shows the size of the use case models constructed using the different guidelines. The fields of the size vector contain the median value (see Section 3.4). The use case models constructed using the Minor guidelines included on average the largest number of actors and use cases, but the lowest number of events in each use case. Only one of the use case models constructed with these guidelines included variations. The use of the Template and Style guidelines resulted in use case models of approximately equal size. However, the use case models constructed with the Style guidelines had slightly more use cases, while those constructed with the Template guidelines included more variations.

Table 7 shows some typical mistakes committed by the subjects in this experiment related to each of the quality attributes.

Table 6. Median size for the use case models

Type of guideline	Size
Minor guidelines	<3,8,2,0>
Template guidelines	<2,5,5,2>
Style guidelines	<2,6,5,1>

Table 7. Typical mistakes in the use case models

Property	Typical mistake(s)
Single diagram	– Splitting the use case model into several diagrams, one diagram for each single use case.
Actors	– Omitting external systems. – Including several actors who have exactly the same goals when using the system and who should therefore have been a single actor.
Use cases	– Including auxiliary functions that were not part of the requirements, overlooking use cases. – Splitting events relating to the same goal on several use cases.
Content	– Omitting assumptions or result.
Level of detail	– Describing what happens inside the system or the user interface. – Giving a brief and too incomplete description.
Realism	– Including common functionality in several use cases when it should have been separated out as a use case on its own. – Omitting variation or neglecting to state where in the flow of events they can occur.
Consistency	– Using different words for the same entity.

5 Discussion

This section discusses how the different guidelines affected the understandability and quality in our experiment.

5.1 Minor Guidelines

The Minor guidelines contained the most elaborate description of how to identify actors and use cases. The use case models constructed using these guidelines included, on average, the largest number of actors and use cases. However, they did not receive the highest score on correct actors and use cases, which indicates that some of their actors and use cases were superfluous. These guidelines also received the lowest overall score on the quality attributes. In our opinion, this indicates that these guidelines gave insufficient support on how to document actors and use cases, because good support for documenting use cases would have helped remove the superfluous actors and use cases.

The groups who used the Minor guidelines seemed to have more problems following the guidelines than the other groups. The use case models constructed using these guidelines did significantly worse on understandability than the use case models constructed with the other guidelines. We believe the reason was that these use cases lacked a coherent structure. The students participating in this experiment also found these guidelines the least useful.

5.2 Template Guidelines

The results in Section 4.1 indicate that use case models constructed using the Template or Style guidelines are easier to understand than use case models constructed using the Minor guidelines. The tests on the scores on correct answers from the customers also give a weak indication that the Template guidelines are better than the Style guidelines. We believe that the structure imposed by the Template guidelines makes it easier to find information in these use case models.

The Template guidelines also did better than the other guidelines regarding different quality attributes of the use case models. In our opinion, this may indicate a relationship between those attributes and the understandability. For example, the groups that followed the Template guidelines handled variations better than the other groups. This is an important aspect since the basic flow of events is often well known, but the alternatives to the basic flow are often not thought of.

These groups followed the guidelines most closely, which indicates that the guidelines are easy to use. The Template guidelines were also considered significantly more useful than the Minor guidelines and slightly more useful than the Style guidelines. We believe that these differences may be due to the templates being easier to understand and apply because it is made explicit what information should be inserted. In our opinion, the support from a template may be particularly important in an environment where the developers have little experience with the application domain and therefore need to work a lot on the requirements to develop a good

understanding. We also believe that the template is particularly useful for developers who have little experience with use case modelling.

5.3 Style Guidelines

Two experiments have previously been conducted to evaluate the Style guidelines regarding how they contribute to completeness, correctness and consistency in the use cases [14,15]. The results from the first experiment showed that the guidelines were usable and helpful in improving the use cases regarding those properties. (These guidelines were proposed by the same research group that conducted the experiment.) However, these results were not confirmed when the experiment was replicated [15], but both agree that the guidelines should be considered when authoring use cases.

In our experiment, the Style guidelines did almost as well as the Template guidelines when the understanding of the readers was compared, and they did better than the Minor guidelines on both quality attributes and usefulness. Hence, it appears that the Style guidelines did better in our experiment than in the experiment reported in [15]. This may be due to a different marking scheme or because the participants in our experiment had more time to thoroughly understand and apply the guidelines. We do not believe that the modifications we made to the original Style guidelines invalidate a comparison, because the changes we made consisted in removing parts of the guidelines that were not applied by the subjects in that experiment. In our opinion, this indicates that Style guidelines may successfully supplement Template guidelines.

6 Threats to Validity

This study is exploratory in the sense that we do not know any other studies where different guidelines on use case modelling have been compared. We would call it a semi-controlled experiment because it was done as part of a course, and thus we had not full control over all parts of the study. For example, we could not control the informal specifications from which the groups constructed the use case models. Moreover, we do not know in details how the students worked when constructing the use case models.

Determining how to analyze the results was a challenge, in particular regarding correctness of the answers to the questionnaires and the different attributes of the use case models. On the other hand, we believe that the organization in customer and developer groups contributed to a realistic setting. The next sections describe factors that we believe may have influenced our results.

6.1 Students as Subjects

Our experiment was conducted with students as subjects. It is therefore uncertain to what extent our results can be generalized to an industrial setting. However, many of the students in this experiment were experienced software developers who work part-time or previously worked in industry. Out of 139 students, 43 answered that they had *some* relevant experience, 9 answered that they had *a lot of* experience. In our

opinion, a large percentage of the students may therefore be comparable to professional software developers with up to two years experience.

Use case modelling is a relatively new technique; many organizations are currently starting to use it. Hence, it is not uncommon that developers have none or only a little experience with use case modelling. So, typical professionals may not have much more experience with use case modelling *per se* than our students, but on the other hand, our students are less experienced with requirements and with modelling in general than are professionals.

Høst *et al.* [20] compared the results from students and professional software developers in a study on factors affecting the lead-time of software development projects. They did not find significant differences between the answers from the students and the professionals, even though knowledge of factors affecting lead-time appears to depend on experience from software development projects.

To overcome the difficulties of using students as subjects, we plan to carry out a similar experiment with 20 professionals as subjects in a four-hour experiment. The explorative nature of this study was an important reason why we chose to conduct an experiment with students as subjects even though they may not be fully compatible with professional developers. Therefore, we found it necessary to test our experimental design on students and perhaps eliminate some hypotheses before conducting an experiment with professionals, as recommended by Tichy [21].

6.2 Complexity of the Task

The use case models constructed in our experiment were smaller than most use case models describing real systems. Therefore, we do not know if our results are applicable when the use case models are considerably larger. We conducted a case study in industry on how to apply use case models in estimation [18]. One of the two software development projects used in that case study was characterized as medium sized. That project lasted 7 months, and the use case model consisted of 7 actors and 9 use cases, which is not substantially more than the number of actors and use cases in the use case models in the experiment reported in this paper. However, the number of events and variations was considerably higher than in our experiment. We therefore believe the use case models in our experiment may be comparable to use case models in small industrial projects.

Our guidelines did not handle secondary actors nor included and extending use cases, and alternative flows of events were handled superficially. This may also mean that the use case models in our experiment are not comparable to use case models for real systems on all aspects. We used quite simple guidelines because we wanted to limit the number of attributes of the use case models that we would have to consider. Again, the reason for this decision was the exploratory nature of our research. Nevertheless, we plan to conduct further studies using extended guidelines.

The differences in size among the use case models in our experiment may have influenced the correctness of the answers about functionality. A small use case model consisting of a small number of use cases described with few details may have made it easy to answer the questionnaire correctly despite low quality of the use model. Correspondingly, very detailed use case models may be time consuming to read,

which in turn may lead to wrong answers even though they are of high quality. Another aspect of this, is that a high quality use case model may have appeared very convincing to the customer group, and thereby led them to believe that all their requirements were included even if they were not.

6.3 Participants both as Customers and Developers

The organization into customer and developers meant that the readers of the use case models all had experience with writing use cases. This will not always be the case for all the stakeholders in a real project. Therefore, we cannot be sure that the result indicating that the use case models constructed using Template guidelines are the easiest to understand, is applicable to customers who are unfamiliar with use case modelling.

However, it is recommended that use case models should be constructed together with their future users. To enable the future users to participate in the use case modelling process, it is common to train them in basic use case modelling. Many customers therefore have some knowledge of use case modelling.

6.4 Motivation

In a large course like our software engineering course, there will inevitably be differences in motivation among the students. We observed differences in motivation regarding answering the questionnaires since the seminars where this was done were compulsory. The students were not used to attendance being compulsory, so some were quite negative about that.

6.5 Dependence on Informal Specifications

The use case models were constructed from informal specifications made by the customer groups. The informal specifications had varying quality, and there were differences in how closely they were followed by the development team. In the cases where the informal specification covers all the information in the use case model, the students may have been able to answer the questionnaire correctly without having understood the use case models. Moreover, although the students were given an explanation of how to answer the questionnaire, some may not have understood exactly on what they should base their answers.

6.6 Experience

Experience with use case modelling may lead to higher quality of the use case models independently of the guidelines. Experience with the application domain might in our experiment have affected the answers to the questionnaires as it may lead to expectations regarding the functionality [22]. However, there were few subjects with application domain experience. We believe that differences in experience did not affect the results since the students were randomly assigned to the groups.

6.7 Questionnaires

Some of the groups made specifications with functionality that was very different from what we expected to be the outcome when we wrote the system descriptions. This lead to some of the questions being irrelevant for some of the use case models. The questions generally seem to have been better suited for system B than for system A. Questions made specifically for each use case model would probably have made it easier to determine how well the use case model was understood. This was infeasible due to the large number of groups and that we wanted the students to fill in the questionnaires shortly after the use case models were completed.

The correctness of the answers was determined subjectively. This may represent a source of error, as it was not always obvious what the correct answer should be. However, the use case models were simple and the answers given were a maximum of two lines of text, so in most cases determining whether an answer was correct was relatively easy.

7 Ethical Considerations

Due to the project being compulsory, the workload on the groups should be approximately even. This meant that all the groups had to use some kind of guidelines, and that learning and implementing them should be equally work consuming. It should also be ensured that all the students had the opportunity to learn equally much.

In our experiment, we attempted to achieve not too large differences between the groups of students by not making the guidelines too different. Of course, this concern made it more difficult to observe the different effects of the various guidelines.

The results from the experiment were presented to the students in a one-hour lecture to give all the students a flavour of all the guidelines. As an afterthought, we realize that this was probably insufficient to ensure that all the students learned equally much. If such an experiment is to be repeated, we would recommend that all the students are given exercises including all the guidelines.

The students were in the questionnaire encouraged to give feedback on how they felt about the experiment. Most of the students were positive, for example, they reported that they through this experiment learned more about use case modelling than they would have done otherwise.

8 Conclusions and Future Work

We have identified three categories of guidelines for use case modelling, and we have conducted an experiment with the aim of detecting the effects of using them. The results from the experiment indicate that guidelines based on templates support the construction of use case models that are easier to understand for the readers than guidelines without specific details on how to document use cases. The guidelines based on templates were also considered by the participants as the most useful when

constructing use cases. Our experiment further indicates that the guidelines based on templates result in better use case models regarding also other quality attributes.

Style guidelines focus on the documentation of the flow of events of each use case. They appear to have contributed to some of the quality attributes. Therefore, it may be beneficial to combine the template guidelines with style guidelines.

This study was exploratory. We will use the results as a basis for further studies both on how to improve the ease of understanding use case models and on how they should be used in subsequent phases of a development project. The following research activities are planned:

- A replication of this experiment using modified versions of the guidelines presented in this paper. The modifications will be based on the results from this experiment and on the extensions suggested in Section 6.2. We also intend to investigate the effects of the different guidelines on the understanding of the groups as a whole, by letting the groups answer the questionnaires instead of the individual participants.
- A follow-up controlled experiment similar to the one reported in this paper, but this time in industry with professional software developers.
- A case study on the application of use case models in software development projects in industry, in particular on how use case models can be used in estimating software development effort.
- A field experiment on how different stakeholders in a project understand use case models and how a reading technique may improve it [22].

Acknowledgements

We thank Lars Bratthall for useful comments on earlier versions of this paper and Erik Arisholm for guidance on the statistics. We also thank all the students who participated in the experiment. This research is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References

1. Jacobson, I. *et al.* Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley, 1992.
2. Kulak, D. & Guiney, E. Use Cases: Requirements in Context. Addison-Wesley, 2000.
3. Booch, G., Rumbaugh, J. & Jacobson, I. The Unified Modeling Language User Guide. Addison-Wesley, 1999.
4. Cockburn, A. Structuring Use Cases with Goals. Technical report. Human and Technology, 7691 Dell Rd, Salt Lake City, UT 84121, Ha.T.TR.95.1, http://members.aol.com/acockburn/papers/usecases.htm, 1995.
5. Cockburn, A. Writing Effective Use Cases. Addison-Wesley, 2000.
6. Schneider, G. & Winters, J. Applying Use Cases – A Practical Guide. Addison-Wesley, 1998.

7. Constantine, L. L. & Lockwood, L. A. D. Software for Use. A Practical Guide to the Models and Methods for Usage-Centered Design. Addison-Wesley, 1999.
8. Rosenberg, D. & Scott, K. Use Case Driven Object Modelling with UML. Addison-Wesley, 1999.
9. Regnell, B., Andersson, M. & Bergstrand, J. A Hierarchical Use Case Model with Graphical Representation. Proceedings of Second IEEE International Symposium on Requirements Engineering (RE'95), York, UK, 1995.
10. Harwood, R. J. Use case formats: Requirements, analysis, and design. Journal of Object-Oriented Programming, Vol. 9, No. 8, pp. 54-57, January 1997.
11. Mattingly, L. & Rao, H. Writing Effective Use Cases and Introducing Collaboration Cases. Journal of Object-Oriented Programming, Vol. 11, No. 6, pp. 77-79, 81-84, 87, October 1998.
12. Jaaksi, A. Our Cases with Use Cases. Journal of Object-Oriented Programming, Vol. 10, No. 9, pp. 58-64, February 1998.
13. Firesmith, D.G. Use Case Modeling Guidelines. Proceedings of Technology of Object-Oriented Languages and Systems – TOOLS 30. IEEE Comput. Soc, Los Alamitos, CA, USA, 1999.
14. Ben Achour, C., Rolland, C., Maiden, N.A.M. & Souveyet, C. Guiding Use Case Authoring: Results of an Empirical Study. Proceedings IEEE Symposium on Requirements Engineering, IEEE Comput. Soc, Los Alamitos, CA, USA, 1999.
15. Cox, K. & Phalp, K. Replicating the CREWS Use Case Authoring Guidelines. Empirical Software Engineering Journal, Vol. 5, No. 3, pp. 245-268, 2000.
16. Hurlbut, R.R. A Survey of Approaches for Describing and Formalizing Use Cases. Technical Report: XPT-TR-97-03, Expertech, Ltd., 1997.
17. Martinsen, S.A. & Groven, A-K. Improving Estimation and Requirements Management Experiences from a very small Norwegian Enterprise. SPI 98 Improvement in Practice: Reviewing Experience, Previewing Future Trends. The European Conference on Software Improvement. Meeting Management, Farnham, UK, 1998.
18. Anda, B., Dreiem, H., Sjøberg, D. & Jørgensen, M. Estimating Software Development Effort Based on Use Cases – Experiences from industry. Submitted to UML'2001 (Fourth International Conference on th Unified Modeling Language).
19. Cioch, F.A. Measuring Software Misinterpretation. Journal of Systems and Software, Vol. 14, No. 2, pp. 85-95, February 1991.
20. Høst, M., Regnell, B. & Wohlin, C. Using Students as Subjects – A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. Empirical Software Engineering, Vol. 5, No. 3, pp. 210-214, November 2000.
21. Tichy, W.F. Hints for Reviewing Empirical Work in Software Engineering. Empirical Software Engineering, Vol. 5, No. 4, pp. 309-312, December 2000.
22. Anda, B. & Jørgensen, M. Understanding Use Case Models. Proceedings of Beg, Borrow, or Steal Workshop, International Conference on Software Engineering, June 5, 2000, Limerick, Ireland.

Appendix A

This appendix contains the three sets of guidelines used in this experiment.

2. Identify use cases and draw a use case diagram

⇒ The use cases describes what the actors wishes to do with the system, that is the actors goals

⇒ Each goal is represented by a use case

⇒ Identify use cases by looking at

- What are the main tasks of each actor?
- Will the actor read/write or change some of the information in the system?
- Should the actor inform the system about changes happening outside the system?
- Does the actor wish to be informed about unexpected changes?

3. Describe each actor

⇒ Give a brief description of each actor with name and most important goals when using the system.

4. Describe each use case in detail

⇒ Describe the flow of events in the use case, that is, all the steps in the interaction between actor and system that are necessary for the actor to reach his goal.

⇒ The description should show:

- The actors input
- Objects (including actors) that are involved
- Assumptions that are made about the objects
- The result of the use case

⇒ Identify other users

- Which interactions will de done with other systems?
- What external hardware is necessary?
- Who performs administration and maintenance?

The second step consists in finding roles:

⇒ Find out what roles the users have (roles encapsulate the way the system is used, but remember that roles are not equivalent to users nor to job-descriptions)

⇒ An actor constitutes a single role

Alternative 2 – Template guidelines

The Use Case Model should include:

1. A use case diagram that shows actors and use cases.
2. A description of each actor according to the first template below.
3. A description of each use case according to the second template below.

Template for describing an actor:

Actor <*name*>	
Description <*A short text that describes the actor*>	
Examples	

Template for describing a use case:

Use Case <*name*>	
Actors <*name*>	
Trigger <*The event which starts the use case*>	
Prerequisites <*Constraints that must be met for the use case to be executed*>	
Post-conditions <*Conditions which are met when the use case terminates*>	
Normal flow of events <*A simple, brief description of the series of events of the most likely outcome*>	
Variations <*Variations on the normal flow of events, why it is followed, and outcome*>	
Use Case associations <*A list of other use cases that this use case is extended by or is used by*>	

Alternative 3 – Style guidelines

The Use Case Model should include:

1. A use case diagram that shows actors and use cases.
2. A description of each actor with name and most important goals when using the system.
3. A description of each use case which shows
    - The actors input
    - Objects (including actors) that are involved
    - Assumptions that are made about the objects
    - The result of the use case

In addition to this each use case should show the flow of events in the use case. The flow of events consists of a number of actions, and each action should be described so that it satisfies the guidelines below.

Style guidelines-

SG1: Write the UC normal course as a list of discrete actions in the form: <action#><action description>. Each action description should start on a new line. Since each action is atomic, avoid sentences with more than two clauses.

SG2: Use the sequential ordering of action descriptions to indicate strict sequence between actions. Variations should be written in a separate section.

SG3: Iterations and concurrent actions can be expressed in the same section of the UC, whereas alternative actions should be written in a different section.

SG4: Be consistent in your use of terminology, that is, use consistent names on actors, objects and actions in all action descriptions. Avoid use of synonyms and homonyms.

SG5: Use present tense and active voice when describing actions.

SG6: Avoid use of negations, adverbs and modal verbs in the description of an action.

Guidelines for content-

CG1: <agent ><action><agent>

CG2: <agent><action><object> <prepositional phrase>

CG3: 'If' <alternative assumption> 'then' <list of action descriptions>

CG4: 'Repeat until' <repetition condition> <list of action descriptions>

CG5: <action 1> 'while' <action 2>

Appendix B

This appendix includes the descriptions of the two systems for which the students should specify a use case model.

Description – System A

An opinion poll institute want a system with questionnaires on the Internet. The system should make it easy to publish questionnaires, as well as to fill in questionnaires on the Web. The answers to the questionnaires should be saved so that they can be exported to other tools (an example is "structured text" which can be imported into a spreadsheet). For some of the questions it will be necessary for the system to read a significant amount of text. The opinion poll institute want an easy overview of the answers received, for example, they want to know how many have answered the different questions. Notice that you shall not make a simple questionnaire on the Web, but a "questionnaire generator" for the Web.

Description – System B

A hospital ward needs a system for swapping duties between nurses. There will be a PC in the ward where the nurses can log in. The user interface should make it possible to register who swaps duties and for what period of time. First the swap is registered with status *inquiry*. When the head of the ward has accepted the swap, the status should be changed to *accepted*. Swaps that are not accepted should be registered with status *not accepted.* If the head of the ward has not responded to the inquiry within 24 hours, status should automatically be set to *accepted.* The nurses must be able to log on to the system to see if the swap of duties is accepted. All accepted duties should be saved so that they can be transferred to other systems.

Appendix C

This appendix shows the questions about functionality in the use case models that were included in the questionnaires given to the students.

Questions for System A:

1. How many questions can there be in a questionnaire? If there is a limit to the number, on what is this limit based?
2. Which different alternatives are allowed for the answers?
3. Is it possible to insert comments either to questions or to answers in questionnaires?
4. Is there any validation of questionnaires that have been completed? If the answer is "yes", how is the validation done?
5. Who has access to the answers from a survey?
6. What possibilities are there in the system for analyzing answers and who has access to these possibilities?
7. To which tool can answers from questionnaires be exported and how is this done?
8. What possibilities exist for changing questionnaires that have already been saved, and who has access to these possibilities?
9. What possibilities exist for changing answers or continue to answer a questionnaire that has already been completed, and who has access to these possibilities?
10. Who has access to publishing questionnaires?

Questions for System B:

1. Who has access to the system and how do they log on?
2. How is the roster made and updated, and who is responsible for it?
3. What possibilities are there in the system to look at rosters and who has access to the different rosters?
4. How is the second nurse (the one who does not initiate the swap) informed that another nurse wishes to swap duties?
5. How is the head of the ward informed about requested swaps?
6. How are the nurses (both the one who initiated the swap and the other) informed about whether a swap has been accepted?
7. Are there possibilities to delete a swap, that is, to return to the original roster?
8. What functionality exists in the system for transferring duties to other systems (for example, the system for paying out wages)?