Understanding Use Case Models

Bente Anda and Magne Jørgensen

Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research, Workshop, 5 June, 2000 at 22nd International Conference on Software Engineering (ICSE), Limerick, Ireland, pp. 94-102, 2000.

Abstract

Use Case Modeling is a technique for handling the functional requirements in a software development project. The Use Case Model can serve as a means of communication between the different stakeholders in a project. It is used in planning the project and is updated and used during the project. In order to reduce the possibilities for misunderstandings and differences in understanding, it would be useful to be able to evaluate to what extent the different stakeholders have understood the model and also to detect differences in interpretation. Low comprehension or differences in interpretation may indicate a need for more effort on specifying the requirements. If this is not feasible, it may be necessary to assume a higher risk when planning and estimating the project. We propose using knowledge on how humans understand text from cognitive psychology in the design of an experiment with a twofold goal: Investigate methods for measuring comprehension of Use Case Models and analyze the differences in understanding.

Keywords: Understanding, Requirements engineering, Use Case Models, Schema Theory

1 Introduction

Requirements engineering is a critical part of the software development process. According to a survey carried out by the Standish Group[1] incomplete requirements are the most important factor in explaining why software projects fail. In addition, problems related to requirements engineering are one cause of many of the other factors that the Standish Group found to be important, for example, changing requirements and specifications as described in [1]. Model clashes are another possible factor in failed software projects [2]. Different stakeholders in a software development project will often have different priorities. These priorities are reflected in different success models of the system. When the models conflict, the project encounters difficulties and may fail.

An increasing number of organizations attempt to improve their requirements engineering process by adopting a particular technique called Use Case Modeling introduced by Ivar Jacobson [3]. The Use Case Model of a system should capture all the functional requirements of the system to be developed and provide a means of communication between the different stakeholders in a development project, i.e. clients, potential users of the system, managers and different groups of developers. Several notations exist [3-6] for Use Case Modelling. The notations differ with respect to the content of the templates that are used for describing the Use Cases and with respect to the degree of formalism. Independently of the notation used, different levels of abstraction are possible, so the requirements engineering team must make decide how much detail they want to put into the specification, both in terms of eliciting and describing the requirements.

There is currently no detailed account of the cognitive processes involved in checking software requirements. While there are several studies on how programmers understand programs [7-14], we have not been able to find any studies on human understanding of Use Case Models. This lack of studies on Use Case Model understanding means that the guidelines and practices on how Use Cases should be described to ease their understandability and provide a basis for a common understanding of the requirements, is highly subjective. A common understanding of the requirements may also reduce the possibility of a model clash. We intend to investigate how understandability of Use Case Models can be measured and how different stakeholders understand Use Case Models. We have recently started projects in two different organisations aimed at investigating this issue

How people understand texts and construct meaning from them, have been studied for several years [15-17]. We therefore believe that knowledge about how humans understand text can provide important insight into how they understand representations of software systems in the form of Use Case Models.

To summarize, the major challenge raised in this paper is:

[1] http://www.standishgroup.com/chaos.html. A study based on a survey of US companies.

Are there findings and methods from the research on how humans construct meaning from text that can be used successfully when studying how different stakeholders read and understand requirements specifications in the form of Use Case Models?
This challenge includes: How can experiences from design of experiments on how humans construct meaning from text be reused in experiments on how requirements specifications in the form of Use Case Models are understood?

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 gives a brief description of how humans understand text. Finally section 4 sketches an experiment for evaluating how different stakeholders understand Use Case Models.

2 Related work

Work has been done on evaluating the understandability of the syntax and semantics of languages for requirements specifications [18]. This work resulted in a set of guidelines for improving understandability of specification languages, applicable to designers of such languages and to specification developers.

Perspective-Based Reading is a technique for detecting defects in requirements specifications [19-21]. The different reviewers of the requirements documents assume different perspectives when inspecting the documents. Several studies have been conducted in order to investigate the effects of this technique compared with checklists or "ad hoc" reading. The results from these studies are not conclusive, and this work differs from ours in that we are primarily interested in understanding, as we believe that to be a prerequisite for detecting errors and handling changes in the requirements.

A set of guidelines for authoring Use Cases have been developed by the ESPRIT 21.903 CREWS research project. These guidelines are intended to improve completeness and correctness and to avoid ambiguities. They were evaluated in [22]. The guidelines give some advice on level of abstraction, but they were not evaluated regarding whether they improve understandability of a Use Case Model.

Much work has been done on program understanding which is similar to understanding a Use Case Model because both include understanding a text expressed with a particular notation. In both cases it is necessary with knowledge of the application domain, and of the syntax and semantics of the language used. There are differences due to programs being expressed in a formal language, while Use Cases are expressed in everyday language, only guided by a template. In addition, programs will only be understood by programmers, while people with a much more varied background should understand requirements. Research on program understanding indicates differences in understanding due to differences in knowledge of the domain, knowledge of the programming language and general problem solving strategy.

There are several cognitive theories potentially useful for describing understanding and complexity. They may be useful to explain why some models are difficult to understand for people, depending on context and background. Examples of such theories are:

- “Chunking theory”. A “Chunk" is a unit of knowledge that has become familiarized and has a place in the memory’s “index”. This theory has, for example, been used to explain performance differences between expert and novice programmers in understanding a piece of program [23].
- “Acceptability principles theory” [24]. According to this theory, difficulties in understanding are caused by the violation of some “acceptability principles”, i.e. commonly accepted rules for how things are to be understood. For example, in [25] the experienced programmers did only outperform the novices when the programs they had to change followed “accepted rules for programming”. If, for example, the program had functions and variable names that did not correspond with the content or function, there was no significant difference between the experienced programmers and the novices.

In addition to these theories there are a lot of other theories and findings within cognitive science and psychology that may be useful for our study. In this paper, however, we have attempted to use Schema Theory [15], a theory of how humans understand text, and we investigate how this theory can be applied to the understanding of Use Case Models.

3 Schema Theory applied to the understanding of Use Case Models

Schema Theory is a frequently used theory in studies on how humans understand text.

Schema Theory was introduced by Bartlett in 1932 [15]. He discovered how people’s understanding and remembrance were shaped by their expectations. He suggested that these expectations could be modeled using schemata.

A schema is defined in [16] as an abstract knowledge structure in the sense that it summarizes what is known about a variety of cases that differ in many particulars.

Schemata are used to make inferences while we read, and inference drawing is a prerequisite for comprehension. The readers create a mental model of the situation and events referred to in the text. A cognitive schema is invoked when we are confronted with a small element of the schema. The schema will then have an important influence on how the rest is understood. Schemata are built from past experience.

A simple example is a schema for "Buying a book on the Internet". If we imagine two stakeholders that are involved in the development of a bookshop on the Internet, the stakeholders may have different schemata for "buying a book on the Internet". One of the stakeholders may have acquired his schema only from buying books in "real" bookshops. The other stakeholder may have experience as a bookseller and a schema acquired from that experience.

The first stakeholder's schema may include the following:

1. Place order
2. Pay
3. Receive order

The second stakeholder's schema may include:

1. Receive information about books
2. Register as customer
3. Place order
4. Receive information about alternative methods of payment
5. Select method of payment
6. Pay
7. Receive order

Due to different experiences in the past, different stakeholders will have developed different schemata and may consequently interpret the Use Cases Model differently.

The effects of different stakeholders using different schemata may result in the following:

- Terms will have different meanings for different people
- Elements are left out because it is believed that they are so obvious that it is not necessary to explicitly include them
- Misunderstandings

Detienne [9] used Schema Theory in the design of an experiment aimed at analyzing the cognitive mechanisms involved in program understanding. She found that the programmers used schemata related to domain knowledge together with schemata related to programming. She found that differences relating to which schema where used in a situation depended on the programmer's experience.

Pressley and McCormick present a number of experiments where people have read stories and found different inconsistencies or have remembered different things according to which schema they used [17]. They also referred to experiments that showed that the participants accepted sentences to be in the text when they were not, but fitted their schema.

We believe that Schema Theory is potentially useful in a study of the understandability of Use Case Models. Schema theory shows that the reader's schemata may be as important as the text itself in determining how a text is understood. The theory stresses the importance of understanding which models the readers use in order to be able to increase their understanding of what is read. These results are important input to our study. Schema theory also allows us to reuse experimental design from other fields, and we will have the possibility to compare our results with existing work. In addition to Schema theory we intend to identify and evaluate other theories and findings within cognitive science and psychology which may be useful for our study.

A Use Case Model consists of a graphical overview, a Use Case Map, and narratives describing each Use Case. We will examine the understandability of both parts in our experiment. Literature on diagram comprehension [26-27] has been searched for theory or methods that could be used in investigating how Use Case Maps are understood. We consider this work less relevant in our context because Use Case Maps use very few symbols and has a very limited syntax compared with most diagram techniques.

4 Experiment

We plan to carry out an experiment with the following two goals:

- The primary goal is to evaluate different experimental designs that can be used in the design of a method for detecting differences in the understanding of a Use Case Model.
- The secondary goal is more exploratory, it is to detect differences in understanding of a Use Case Model, differences in the schemata that are used and possible reasons for these differences, and to get indications regarding how notation and level of detail in the Use Case Model influence understanding.

This section describes the first version of an experimental design. The design is inspired by experiments on understanding text referred to in Bartlett [15] and Pressley and McCormick [17] and by experiments on program understanding performed by Von Mayrhauser and Lang [7] and Detienne [9]. The design of the experiments on Perspective-Based Reading [20] will also be used to further elaborate this design.

Hypothesis

- It is possible to find an optimal level of detail for the Use Case Model for a specific purpose and under given circumstances.

Subjects

The subjects will be different stakeholders in a software development project.

There will be approximately 20 subjects. The subjects will have varying experience and knowledge about the application domain, and varying experience and training in Use Case Modeling.

Artifacts

The subjects will be presented with Use Case Models describing the functional requirements of two small applications. The following variations of Use Case Models will be used.

- Use Case Maps with highlevel Use Case templates.
- Use Case Maps with relations between Use Cases and expanded Use Case templates with scenarios.

The first variant represents a typical description of a Use Case Model developed very early in the project, while the second variant is more elaborate.

Method of investigation

The primary goal of the experiment is to find a method that can be used to assess comprehension and detect misunderstandings and differences in interpretation. Several methods will be used and evaluated. A combination of methods may be used for each subject. The methods are:

- *Think-aloud protocol [7]*. The subjects are encouraged to say what they are thinking when reading and understanding the Use Case Model.

- *Questions [9]*. After reading the Use Case Model the subjects respond to questions. The questions should indicate comprehension and ability to use information extracted from the Use Case Model.
- *Reproduction [15]*. The subjects will be asked to describe the Use Case Model in their own words after having read it.
- *Inconsistencies and lack of precision in the Use Case Model [17]*. Deliberate inconsistencies and lack of precision will be introduced in the Use Case Model. The experiment will investigate, through questions after the subjects have read the Use Case Model, whether there are differences in the inferences made when information is lacking, and if there are differences in how many errors/inconsistencies are found by the different stakeholders.

For all methods, the goal is to detect differences in the schemata, the search strategies and differences regarding which inferences are made.

Threats to validity

The following may represent threats to the validity of the results:

- Few subjects mean that they will study the Use Case Models individually. This situation may differ from how they usually read requirements documents. The limited number of subjects may also make it difficult to obtain statistical validity.
- Each subject will investigate several Use Case Models. There may therefore be learning during the experiment, which may affect the result.
- The Use Case Models used in the experiment will describe the functional requirements of two small applications. These Use Case Models may not be representative of other, larger applications.

Acknowledgements

We gratefully acknowledge the support from our industrial partner, Icon Medialab, in particular Alexander Woxen and Per Einar Dybvik. The research project is funded by The Research Council of Norway through the industry-project PROFIT (PROcess improvement For the IT industry).

References

1. Pfleeger, S. L. *Software Engineering. Theory and Practice.* Prentice Hall Inc., 1998.
2. Boehm, B. & Port, D. "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them". Software Engineering Notes, vol. 24, no.1, 36-48, 1999.
3. Jacobson, I. et al. *Object-Oriented Software Engineering. A Use Case Driven Approach.* Addison-Wesley, 1992.
4. 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.

5. 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.
6. 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.
7. Von Mayrhauser, A & Lang, S. "A Coding Scheme to Support Systematic Analysis of Software Comprehension". IEEE Transactions on software Engineering, vol. 25, no. 4, 526-540, 1999.
8. Brooks, R. "Towards a Theory for the Comprehension of Computer Programs". International Journal of Human-Computer Studies, vol. 51, nr. 2, 197-211, 1999.
9. Detienne, F. "Une application de la théorie des schémas a la comprehension de programmes". Le Travail Humain, tome 51, no. 4, 335-350, 1988.
10. Adelson, B. “Problem solving and the development of abstract categories in programming languages”. Memory and Cognition, vol. 9, no. 4, 422-433, 1981.
11. Miara, R. et al. “Program indentation and comprehensibility”. Communications of the ACM, vol. 26, 861-867, 1983.
12. Gilmore, D. J. and Green, T.R.G. “Comprehension and recall of miniature programs”. Journal of Man-Machine Studies, vol. 21, 31-48, 1984.
13. Letovsky, S. and Soloway, E. “Delocalized plans and program comprehension”. IEEE Software, vol. 3, no. 4, 41-49, 1986.
14. Linos, P. et al. “Facilitating comprehension of C-programs: an experimental study”. Proceedings IEEE Second Workshop on Program Comprehension, 55-63, 1993.
15. Bartlett, F. C. *Remebering. A Study in Experimental and Social Psychology*. Cambridge University Press, 1932.
16. Anderson, R.C., & Pearson, P.D. *A schema theoretic view of basic processes in reading*. P.D. Pearson (Ed.) Handbook on reading research. New York, Longman, 1984.
17. Pressley, M. & McCormick, C. B. *Advanced Educational Psychology for Educators, Researchers and Policymakers*. Harper Collins, 1995.
18. Britton, C. & Jones, S. "The untrained eye: how languages for software specification support understanding in untrained users". Human-Computer-Interaction, vol. 14, nr.1-2, 191-244, 1999.
19. Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. & Zelkowitz, M." The Empirical Investigation of Perspective-Based Reading". Empirical Software Engineering Journal, vol 1, nr. 2, 133-146, 1996.
20. Basili, V.R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sørumgård, S. & Zelkowitz, M. *Lab Package for the Empirical Investigation of Perspective.Based Reading*, 1998. Available at: http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/manual.html.
21. Regnell, B., Runeson, P. & Thelin T. "Are the Perspectives Really Different? - Further Experimentation on Scenario-Based Reading of Requirements", Technical Report CODEN: LUTEDX(TETS-7172)/1-40/1999, Dept. of Communication Systems, Lund University, 1999.
22. 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.
23. Schneiderman & Mayer. “Syntactic/semantic interactions in programmer behaviour: a model and experimental results”. International Journal of Computer and Information Sciences, vol. 8, 219-238, 1979.
24. Kosslyn , S. M. “Understanding charts and graphs”. Applied Cognitive Psychology, 3, 185-223, 1989.

25.E. Soloway, B. Adelson, and K. Ehrlich, “Knowledge and process in the comprehension of computer programs,” in *The nature of expertice*, M. T. H. Chi et al., Eds.: Lawrence Erlbaum Assoc.,129-152, 1988.
26.Winn, W. "An Account of How Readers Search for Information in Diagrams". Contemporary Educational Psychology, vol. 18, nr. 2, 162 - 185, 1993.
27.Glasgow, J., Narayanan, N.H. & Chandrasekaran, B. *Diagrammatic reasoning : cognitive and computational perspectives*. Menlo Park, Calif.: AAAI Press, 1995.