INTEGRATION OF REQUIREMENT ENGINEERING WITH UML IN SOFTWARE ENGINEERING PRACTICES

Abu Buker Siddique[1], Salman Qadri[1], Shafiq Hussain[2], Shabir Ahmad*[3], Imran Maqbool[1] and Abdul Karim Nawaz Khan[1]

[1]Department of Computer Science, Islamia University Bahwalpur, Pakistan.
[2]Department of Computer Science, Bahauddin Zakariya University, Sub-Campus Sahiwal, Pakistan
[3]Government College of Commerce, Multan, Pakistan
Email: mian_shabbir@hotmail.com (Corresponding Author)

ABSTRACT--*The Unified Modeling Language (UML) is now the de facto modeling language widely adopted in software engineering. UML is used for developing adaptive and customizable systems on the basis of particular requirements. The main objective of UML practices is to achieve a requirement engineering process in a comprehensive manner. This paper explores how Use Case can be applied in different project by the team and what types of problems may a software company face during the implementation. This study is conducted by surveying sixty six numbers of software companies. On the basis of survey and literature study, improvement suggestions for the Use Case are given in this paper. These suggestions take into account the different uses of Use Cases in a project as well as the phase in which Use Case is used. The results of this survey ascertain that Use Cases practices are 89% useful for developers and 97% for clients point of view. The most important survey's findings are: (1) that during discussions with clients, Use Cases should be supplied to the user along with interface prototypes, (2) that companies should make use of related Use Cases and (3) that in Use Cases, user interface has no need of details.*

KEYWORDS: Requirement Engineering, Requirement Specification, UML, Use Case, Software Companies.

1. INTRODUCTION

The concept of Use Cases was first introduced by Ivar Jacobson in 1987 as a tool for modeling functional requirements, Jacobson, 2004[1]. The idea was quickly adopted world-wide as the book Object-Oriented Software Engineering was published. A Use Case Driven Approach was introduced by Jacobson et al. 1992 [2] and has remained an important method for requirements management. Cockburn, 1997 [3], stated simply how an actor interact with a computer system to achieve a goal.

A Use Case can be written either by the client, the developer or by the client and developer as a team. Use Cases connect many other requirements details and provide scaffolding that connects information in different parts of the requirements. They are connected to other requirements as user interface requirements, user interface details and business rules, Cockburn, 2001[4].

Regnell et al. developed an approach where use cases are used to capture requirements [5]. A use case is the specification of a sequence of actions, including variants that a system can perform, interacting with actors of the system [6]. Use cases have become one of the favorite approaches for requirements capture more so ever since their adoption by software development approaches such as the Unified Process [7]. Nebut, Clementine, et al. [8], introduced an automatic use case driven approach in 2006. The generated state machines are used as prototypes for requirements validation by simulation. Because of the automated generation, prototypes in the approach are obtained from requirements in a timely manner with little effort. However, initial applications of the approach brought some requests for improvement from the users of the approach.

One of these requests is to provide a mechanism that would facilitate repeatability of simulation sessions. There exist two principal Use Case notations, i.e. as textual descriptions and as graphical representations. In this paper we will call the textual notation as Use Case descriptions, the graphical representation as Use Case diagrams, and we will use the term Use Cases as a collective name for both.

2. BACKGROUND

Previous research has investigated the comprehensibility and application of Use Cases. Some research has been conducted in a real software development setting while other research has been conducted on students.

2.1 Previous Research in the Software Industry

This section describes the findings of a survey that identified how Use Cases are employed in practice are described.

2.1.1 A Survey of Use Cases in Practice

In Use Cases in Practice, the authors attempt to find out how Use Cases are employed by developers. The most important result from this survey was that industrial practices place emphasis on the coupling between Use Cases and user interface details even though this is not recommended. The authors suggest using task models as a complementary to Use Cases. A task model specifies what the user does, or wants to do, and why, and is similar to Use Cases. In contrast to Use Cases the tasks in task models are decomposable into subtask and atomic actions. Based on the task model, the user interface may be automatically generated. Another issue was that the participants in this survey had problems modeling and understanding the «include», «extend» and generalization relationships, Sinnig et al., 2005[9].

2.1.2 Problems in Real Projects Using Use Case Diagrams

This section describes, how to avoid Use-Case pitfalls, Susan Lilly[10] focuses on problems with Use Case diagrams based on observations from a number of real projects. The top ten problem using Use Cases and their solutions as describes in the table 1.

Top ten problems from real projects	
Problem 1:	The system boundary is undefined or inconsistent.
Cure:	Be explicit about the scope, and label the system boundary accordingly.
Problem 2:	The Use Cases are written from the system's point of view.
Cure:	Name the Use Cases from the perspective of the Actor's goals.
Problem 3:	The actor names are inconsistent.
Cure:	Get agreement early in the project and establish a glossary to define the actors
Problem 4:	Too many Use Cases.
Cure:	Make sure the granularity of the Use Cases is appropriate.
Problem 5:	The actor-to-use-case relationships resemble a spider's web.
Cure:	The actors should not be defined too broadly.
Problem 6:	The Use Case specifications are too long.
Cure:	The granularity of the Use Cases may be too coarse.
Problem 7:	The Use Case specifications are confusing.
Cure:	Include a context field in your Use Case specification template to describe the set of circumstances in which the Use Case is relevant.
Problem 8:	The Use Case does not correctly describe functional entitlement.
Cure:	Make sure that each actor associated with a Use Case is completely entitled to perform it. If an actor is only functionally entitled to part of the Use Case, the Use Case should be split.
Problem 9:	The client does not understand the Use Cases.
Cure:	Teach them just enough to understand. Put a short explanation in the document, lead a training course and think long about using <<include>> and <<extend>>.
Problem 10:	The Use Cases are never finished.
Cure:	Loosely couple user interface details and Use Case interactions.

Table 1. Problems from real projects

2.2 Empirical Research Conducted on Students

Several researchers have conducted experiments on students to study the Use Case technique.

2.2.1 Other Researcher's Empirical Research

Anna Bobkowska has reported some problems regarding Use Case diagrams. Among these were the stick-man notation that are not intuitive to use for representing computers and that the direction of the «extend» and «include» arrows are confusing, Bobkowska, 2005[11].

In Quality and Understandability of Use Case Models, the authors perform an experiment on students with the aim to detect effects of guidelines when writing Use Cases. The result from this experiment indicated that guidelines based on templates constructs Use Cases that are easier to understand than guidelines without specific details on how to document each Use Case, Anda et al., 2001[12].

3. METHODOLOGIES OF REQUIREMENT ENGINEERING

Different methodologies are used in requirement engineering as follows:

3.1 Use Case Brief

The most important advantage of Use Cases is that they describe a system in a manner that all stakeholders can understand. They are therefore used as a contract between stake-holders for the behavior of the computer system, Cockburn, 2001[4]. As a user-centered technique, Use Cases capture the requirements from the user's point of view, ensuring that the correct system is developed. Other benefits of using Use Cases are their usefulness in estimating, scheduling and validating effort, and that test cases can be directly derived from them. Use Cases contain a description of things that might go wrong, and projects benefit from having exceptions identified early because it saves time later in the project, Firesmith, 1995[13].

3.2 Casual Form

An informal way of writing a Use Case is as a narrative, called Casual form. The Use Case is written in prose and describes at a high level how an actor interacts with the system to accomplish a goal.

3.3 UML Use Case Diagrams

The Use Case formats described so far, are all versions of Use Case descriptions. Now we describe UML Use Case diagrams in detail [14, 15, 16]. UML Use Case diagrams consist of actors and use cases (ellipses) which are connected by a link or a specific relation. The most common relations, in addition to the normal links, are «include», «extend» and generalization. Figure 1 illustrates a Use Case with normal links and the «include» and «extend» relation.

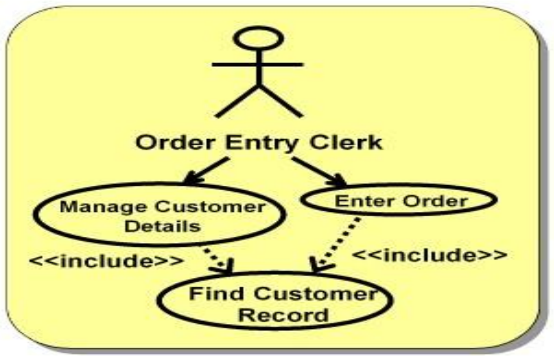

Figure 1: Use Case Diagram

3.4 Other Requirements Engineering Techniques

This section gives an overview of two other techniques used in requirements engineering. These techniques are User stories and SRS.

3.4.1 User Stories

A user story describes functionality that is valuable to a user of a system. User stories were first introduced in eXtreme Programming (XP) as a way of expressing requirements. The story is hand-written on a card. This card is the visible manifestation of the User story, but the conversation where the details are worked out is the most important.

3.4.2 Software Requirements Specification

IEEE has developed a standard for how to write a good Software Requirements Specification (SRS) [17, 18] In this standard the system is described as a set of "The system shall ...."-sentences that focus on what functions the system shall support, in contrast to Use Cases that describes how the system is used by a user. Writing requirements compliant to this standard often result in tedious and boring reading. This might result in the specification not being read carefully enough, and the project team not getting enough information about the requirements from the client, Cohn, 2004[19]. The IEEE framework for the requirements specification is especially appropriate in classic models of the software development process; the waterfall model and its variants, Vliet, 2000[20, 21, 22].

4. OPERATION OF USE CASE

Operations of Use Case are for operation of the surveys. These operations are used for the preparation and execution of the surveys.

4.1 Operation of the Survey

This section describes the preparation, execution and data validation of the survey.

4.1.1 Preparation

Before sending the survey out, two persons read through the survey to reveal vagueness in the question wording. We received a list of company names from a students. This list was used to find contact information on the Internet. We emailed this survey about sixty six companies.

4.1.2 Execution

The survey was written in Microsoft Word, so the respondents had to fill out the Word document and email it back to us. We received thirty eight replies from sixty six companies. We do not know the exact time they spent on it, but we believe it took approximately ten to fifteen minutes to answer, depending on how much complementary text each person wrote.

5. DESIGN OF THE SURVEY

This section describes how the survey is related to the research questions and how it meet the survey objectives. The survery objectives and the survery questions are describe in the table 2.

Survey Objectives	
Investigate how Use Cases are applied in the software industry and how well the technique works for different purposes. In addition, investigate how the Use Case technique can be improved based on developers and clients experience with Use Cases.	
Survery Questions	
RQ1	When is it appropriate to apply Use Cases?
RQ2	For what purposes are Use Cases applied?
RQ3	How well did Use Cases work in a specific project?
RQ4	Do Use Cases work well in discussions with clients?
RQ5	What is difficult and what is simple about Use Cases?
RQ6	How can we improve the Use Case technique?

Table 2 presents the survey objectives and survey questions

5.1 Personal Information and Information about the Company

The first part focused on personal information and information about the company. The respondents were asked to fill in their name, although this was optional. They were also asked to fill in the name of the company, their position in the company, and the number of years spent in the industry.

5.2 Application of Use Cases in the Company

This part was related to RQ1 When is it appropriate to apply Use Cases? and RQ2 For what purposes are Use Cases applied?. The first question was an open question about what other responsibilities the person(s) who write Use Cases have. In the survey the respondents were asked whether they would have preferred a specialized tool for Use

Cases. Then they were asked to check for client representatives, developers or testers to the question on what persons Use Cases were primarily written for. If the respondents had written Use Cases him or herself, he or she was asked to describe how they proceeded when writing them. The last question in this part was about how many pages an average Use Case is.

5.3 Use Cases as a Tool for Communication with the Client

This part was related to RQ4 Do Use Cases work well in discussions with clients? On a scale from one, I strongly disagree, to five, I strongly agree, the respondents were asked to consider seven assertions about how appropriate Use Cases are as a tool for communication with client representatives.

5.4 Personal Opinions about Use Cases

This part was related to RQ1 When is it appropriate to apply Use Cases? and RQ5 What is difficult and what is simple about Use Cases? The respondents were asked to consider eight assertions about things that are difficult about Use Cases on a scale from one, I strongly disagree, to five, I strongly agree. Then they were asked to consider on the same scale in what types of projects Use Cases should be used: development of completely new systems, further development of old systems, Internet portal projects, and projects where one works near the client representatives. The last section of this part focused on how many times the respondent had experienced problems with Use Cases: I have experienced that we have spent so much time on the Use Cases in the beginning of a project that when the development started, the Use Cases have become irrelevant, I have experienced that Use Cases have become so extensive that I did not bother to read through the whole Use Case, and so on. The alternatives to these questions were Never experienced, Experienced once and Experienced two times or more.

5.5 Personal Experience with Use Cases

Part five was related to RQ1 When is it appropriate to apply Use Cases? and RQ5 What is difficult and what is simple about Use Cases. It contained two open questions. The first was: Please describe some positive experience with Use Cases, and the second was: Please describe some negative experience with Use Cases.

5.6 Improvement Suggestions to the Use Case Technique

Part six was related to RQ6 How can we improve the Use Case technique?. On a list of five improvement suggestions for the Use Case technique, the respondents were asked to check the suggestions that they thought would have been useful.

6. RESULTS FROM THE SURVEY

This section presents the results from the survey. The results are presented according to the research question where they belong.

6.1 General Information about Use Cases

The respondents were asked whether they write Use Case diagrams, Use Case descriptions or a combination of both. Figure 2 shows the distribution of answers.

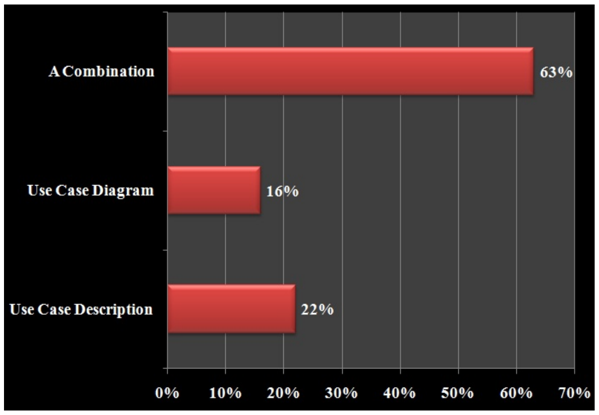

Figure 2. Types of Use Cases applied

All participating companies seemed to have similar procedures for writing Use Cases. They arrange meetings or workshops with the client or the users where the primary goal is to find the actors and goals. Based on this information an overall Use Case diagram is written and from this diagram all Use Case descriptions are identified and prioritized.

6.2 Summary of Results

Table 3 presents the most important results for each survery question.

RQ1 When is it appropriate to apply Use Cases?	
89%	Agreed that it is appropriate to apply Use Cases when developing completely new system.
64%	Agreed it is appropriate to apply Use Cases when developing Internet portals.
RQ2 For what purposes are Use Cases applied?	
86%	Apply Use Cases for structuring the requirements specification.
86%	Use it for estimation.
82%	Use it for programming.
68%	Use it for creating test cases.
RQ3 How well did Use Cases work in a specific project?	
94%	Testers thought the Use Case descriptions worked well.
RQ4 Do Use Cases work well in discussions with clients?	
97%	Agreed that a prototype of interfaces should be used in addition to Use Cases, because the client needs a picture of what the system is going to look like.
64%	Agreed that Use Case is a good technique for communication with clients that are not so familiar with IT-technology.
75%	Disagreed to the assertion that clients think Use Cases are futile.
RQ5 What is difficult and what is simple about Use Cases?	

66%	Agreed that it is difficult to find the right level of detail that suits developers, tester and the client.
61%	Disagreed that it is difficult to write Use Cases.
RQ6 How can we improve the Use Case technique?	
71%	Agreed that it would be useful with a tool that makes it easier to get an overview of related Use Cases.
63%	Agreed that it would be useful with a standardization of layout and content in the Use Cases.

7. RESULTS AND DISCUSSIONS

This section discusses the results that we got from our surveys and relates them to previous research. Based on the results we conclude with a list of suggestions of how the Use Case technique can be improved.

7.1 When is it appropriate to apply Use Cases?

Result: When developing completely new systems.
Discussion of result: According to our results, Use Cases are appropriate to employ when developing completely new systems, and not that appropriate when developing further on existing systems. The reason for this is that for existing systems much of the specification is already written down and making Use Cases would be superfluous.

7.2 For what purposes are Use Cases applied?

Result: For structuring the requirements specification, estimating, programming and constructing test cases. Less commonly, it is also used for system documentation, writing user documentation, preparing courses and testing directly from Use Cases.
Discussion of results: The results show that most of the companies employ Use Cases for structuring the requirements specification and for estimating projects. Many also use it for programming and to create test cases.

7.3 How well did Use Cases work in a specific project?

Result: For the purpose of testing from the scenarios in the Use Case descriptions, the testers thought the Use Case descriptions worked well, but the Use Case descriptions became long and too detailed according to the developers.
Discussion of result: We got the impression that this way of applying Use Cases worked well in this project. The project leader stated that she was pleased with the implementation of this particular project. The reason for the success was that one person had full responsibility for the Use Case descriptions. The testers were also pleased with the Use Cases because they were easy to read and made it easy to know what to test. A drawback with this use was that the Use Case descriptions became long and hard to keep updated.

7.4 Do Use Cases work well in discussions with clients?

Result: Yes, if used together with user interface prototypes and written in a language that the client understands.

Table 4. Suggestions for how to improve Use Cases

Suggestions for how to improve Use Cases	
Suggestion 1:	Make use of a tool that makes it easier to get an overview of related Use Cases and other documents.
How to:	Create a Wiki where you relate interfaces, business rules, Use Cases etc.
Suggestion 2:	Do not emphasize too much on writing extensive Use Cases in the beginning of the project.
How to:	Start with Use Case briefs or Casual form, and expand the Use Cases into fully dressed versions later in the project.
Suggestion 3:	Standardize your Use Cases.
How to:	Set clear directions for the layout and the content of the Use Cases.
Suggestion 4:	Avoid unnecessary changes in the Use Cases when the user interface design changes.
How to:	Do not write details about the user interface in the Use Cases. Instead, create a low fidelity prototype of the interface in addition to the Use Case.
Suggestion 5:	Maintain the Use Cases throughout the project.
How to:	Make sure one person has the full responsibility for updating and maintaining the Use Cases.
Suggestion 6:	Use terms in a consistent way.
How to:	Create a glossary of terms for the requirements document that is used in a consistent way.
Suggestion 7:	Make sure your client understands the Use Cases properly.
How to:	Provide a user interface prototype in addition to your Use Cases and provide training courses for your clients.
Suggestion 8:	Get more than one point of view when writing Use Cases.
How to:	Make sure at least two persons write each Use Case.

Discussion Of Result:

The opinions on Use Cases as a tool for communication are divided. Some developers think that Use Cases should be used primarily as a tool for communication, while others believe Use Cases should not be used for this purpose at all. Either way, the most evident result of this study is the following: A prototype of the inter-face should be used in addition to the Use Case when you communicate with the client. This does not, however, imply that prototypes can substitute Use Cases.

7.5 How can we improve the Use Case technique?

Based on the results from this study, we have made a list of suggestions for how to improve Use Cases in Table 4.

8. CONCLUSION

Use Case plays an increasingly important role in requirement engineering and software development. Based on a number of industrial survey and case studies, we have proposed different suggestions. These suggestions take into account the different uses of Use Cases in a project as well as the phase in which Use Case is used.

We relate these Use Cases practices for developers and clients point of view, although it is hard to get clear, precise answers to the question about in what projects Use Cases are appropriate to use. The answers were quite superficial: Use Cases are appropriate to apply in large projects, web projects and in projects where completely new systems are developed.

Recently, research has been carried out to investigate how UML can be used in the testing phase of the software development process. As a result, a number of UML-based coverage criteria have been proposed in the literature. These criteria are based on coverage of elements of UML diagrams. In this paper, these UML-based criteria are surveyed. For each of the criteria a formal definition is presented which is necessary to facilitate a comparison of the criteria. It has been found that relatively little work has focused on empirically investigating how effective the criteria are at detecting faults. Furthermore, no research has been carried out to show how the various criteria relate to each other in terms of the coverage they provide. Therefore, it is believed that these are important topics for future investigation.

REFERENCES

1. Jacobson, Ivar. "Use cases–Yesterday, today, and tomorrow." *Software and Systems Modeling Springer* **3**(3): 210-220(2004).
2. Jacobson, Ivar. *"Object oriented software engineering: a use case driven approach."* (1992).
3. Cockburn, A.: 1997, Structuring Use Cases with goals, *Journal of Object-Oriented Programming, SIGS Publications,* Nov-Dec 1997.
4. Cockburn, A.: 2001, Writing effective use cases, *Addison-Wesley*.
5. Regnell, Björn, Kristofer Kimbler, and Anders Wesslen. "Improving the use case driven approach to requirements engineering." In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on,* pp. 40-47. IEEE, (1995).
6. Dumas, Marlon, and Arthur HM Ter Hofstede. "UML activity diagrams as a workflow specification language." *≪ UML≫ 2001—The Unified Modeling Language. Modeling Languages, Concepts, and Tools*. Springer Berlin Heidelberg, 76-90(2001).
7. Jacobson, G. Booch, and J. Rumbaugh. “*The Unified Software Developm ent Process”*Addison Wesley, 1998.
8. Nebut, Clementine, et al. "Automatic test generation: A use case driven approach." *Software Engineering, IEEE Transactions on* **32**(3): 140-155 (2006).
9. Sinnig, D., Rioux, F., and Chalin, P.: 2005, “Use Cases in Practice: A survey”, *http: //www.dsinnig.com/pdfs/CUSEC05_Sinnig.pdf*
10. Lilly, S.: 2001, How to Avoid Use-Case Pitfalls, *Software Development Magazine*
11. Bobkowska, A.: 2005, A Methodology of Visual Modeling Language Evaluation, *Springer-Verlag Berlin Hedelberg* 2005.
12. Anda, B., Sjøberg, D., and Jørgensen, M.: 2001, “Quality and Understandability of Use Case Models”, *Springer-Verlag*.
13. Firesmith, D. G.: 1995, “Use Cases: the Pros and Cons”, *ROAD*, **2(**2): 2-6(1995).
14. Regnell, Björn, Kristofer Kimbler, and Anders Wesslen. "Improving the use case driven approach to requirements engineering." In *Requirements Engineering, 1995., Proceedings of the Second IEEE International Symposium on,* pp. 40-47. IEEE, 1995.
15. Van Lamsweerde, Axel. "Goal-oriented requirements engineering: A guided tour." In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on,* pp. 249-262. IEEE, 2001.
16. Azam, Farooq, et al. "Framework Of Software Cost Estimation By Using Object Orientated Design Approach." *IJSTR* **3**(8): 97-100(2014).
17. Porter, Adam A., Lawrence G. Votta Jr, and Victor R. Basili. "Comparing detection methods for software requirements inspections: A replicated experiment." *Software Engineering, IEEE Transactions on* 21.6.1995: 563-575(1995).
18. Baloch, Muhammad Perbat, et al. "Comparative Study of Risk Management in Centralized and Distributed Software Development Environment." *Sci.Int.(Lahore)*, **26**(4), 1523-1528(2014).
19. Cohn, M.: 2004, Advantages of User Stories for Requirements, *Prentice-Hall*
20. Vliet, H. V.: 2000, Software Engineering, Principles and Practice, *Wiley*
21. Ahmad, Shabir, and Bilal Ehsan. "The Cloud Computing Security Secure User Authentication Technique (Multi Level Authentication)." *IJSER* **4**(12): 2166-2171 (2013).
22. Nuseibeh, Bashar, Jeff Kramer, and Anthony Finkelstein. "A framework for expressing the relationships between multiple views in requirements specification." *Software Engineering, IEEE Transactions on* 20.10 (1994): 760-773.