Improving Software Requirement Elicitation using Q-Use Case

Sakthi Kumaresh
Dept. of Computer Science
M.O.P. Vaishnav College for Women, Chennai, India

S. Sruthi
Dept. of Computer Science
M.O.P. Vaishnav College for Women, Chennai, India

ABSTRACT
Requirement elicitation has been one of the most challenging aspects of software development. Not only complete requirements are difficult to gather at the beginning of a project, they tend to change and widen along with the duration of the project. In addition the largest number of defects that remain in delivered software is attributable to defective requirements. They are also very costly to correct, which can be as much as several orders of magnitude more than those introduced in the earlier stages of software development. A use case describes how someone or something would interact with proposed system. Use cases are commonly used as a tool during requirements engineering [3]. This paper presents a systematic approach to eliciting quality requirements based on use cases, with emphasis on building the right product. The approach extends traditional use cases to also cover Q-Use Case, and is potentially useful for checking against the 4 C's – Correctness, Completeness, Clarity and Consistency in order to strengthen requirement elicitation and to achieve high quality in software development.

Keywords
Requirement Elicitation, Use Case, Defects, Requirement Engineering.

1. INTRODUCTION:
The greatest challenge to any thinker is stating the problem in a way that will allow a solution. In software development these words are more apt from requirements elicitation point of view. The engineering of requirements may be the most important activity of the software development life cycle. This is because requirements ultimately define what developed systems will be like and possibly, how they will be developed. This paper proposes a new use case called Q-Use Case (QUC) apart from normal use case that are used in use case diagram. QUC does the verification of use cases by following quality use case checklist against the use cases

1.1 Use case
A use case depicts flow of events of a particular system to be proposed and is represented in form of its own descriptive diagrams. Use case methodology is used to identify, clarify and organize system requirements. Each use case describes the interactions between the system and the user (human). The use case technique consists of a use case model and a method to create use case models. The use case model consists of use case diagrams and several use case descriptions. Use cases also indirectly convey how the users should interact so that the system will be able to perform its intended function. An actor constitutes set of roles that users can perform when interacting with the system [8]. Use cases are used in a use case diagram to show the relationship with the system or entity or their actors. Use cases act as bridges between user, requirement and implementation of the system. Constructing a use-case model includes three stages:

- Identify the original(actual) use cases to capture minimum requirements,
- Identify extension use cases to construct a more complete model than before.
- Refine a use case model to enhance reusability

1.2 Use case approach:
In use case approach, an actor is a representation of all users who interact with the system in a given role. The goal of the use case approach is to describe realistic scenarios for using the features defined in the requirements. The relationships *uses* and *extends* between use cases in use case approach specifies how one use case may be embedded into another one, extending its functionality. For example, a Location Based Services System use case might look like Fig 1. A use case gathers functional requirements of the system. Functional requirements describe what the system should do whereas Non functional requirements describe how the system will do it. Non functional requirements are not covered under use case modeling [5]. There is no absolute scale consideration of a non functional requirement.

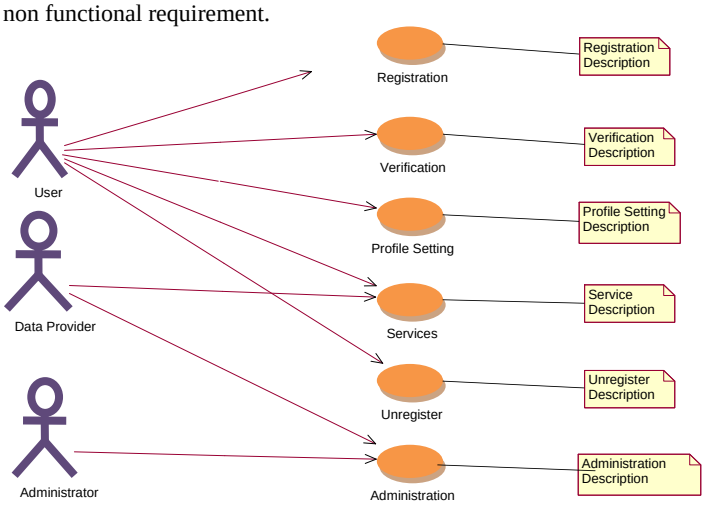

Figure 1 Use Case diagram for Location Based Services System

1.3 Why consider use cases for requirement elicitation?

The most complex way while creating sophisticated software systems is to "*build the right product"*. Projects that is incomplete and difficult to understand the requirement specifications of the system end up with scopelessness, expensive rework, exceeding the time limit, or even abandonment. Hence, this study focuses on flaw free requirement elicitation approach, by way of integrating Q-Use Case along with normal use case approach.

2. LITERATURE REVIEW:-

A use case does not have variation in the family of products. Many experience that the decision on abstractedness is arduous in which taking the use cases is out of control and propagation in their use cases. The core functions and features mentioned in many papers are more brief. Non-functional requirements do not require any function to be performed by the system. They are not limited to any aspects of controlling the requirements elicitation as they do not require the basic functionalities. The requirements which are discovered to enlighten the path of software development form the basis of the quality of the system. Many teams tend to cause utile decay by identifying the use cases closely [14]. The menace is that system will not be very stretchable. The papers presented in IEEE and ICJA mainly focuses on qualities like trustworthy, the time factor, renown and tentative. The models mentioned under the software development uses techniques like use cases and activity charts which helps to discover the requirements and functionalities of the system, points the intrinsic behavior of the essence and software developed.

The Reasons mentioned for why using use cases for requirements gathering are [16]:-

- They deed the development process
- They are easily understood and read.
- They can recognize the element reprocess
- They can be used to grade needs

In some papers in order to improve the levels of understanding use cases with the support of use case modeling they derive the understanding visually by viewing the picture. Many methodologies are mentioned by the papers in order to provide the needs and face the challenges to build huge and complex projects, tools are used for the use of semantic web technologies and to store the information, the branch of Meta physics (ontology) present common reference for the creators of software engineering [13].They use requirements engineering to support use cases where the requirements are based on system, user, functional, non-functional. The use cases must be validated by themselves. Innovatory solutions cannot be contemplated. Use case doesn't cover some software aspects. Many approaches like to derive use cases from event table have been taken in order to speed up production of use case diagrams. The literature review for this thesis was gathered from many research papers.

3. PROBLEMS FACED IN REQUIREMENT DEFECTS:

- Ambiguous understanding of processes
- Inconsistency within a single process by multiple users
- Insufficient input from stakeholders
- Conflicting stakeholder interests
- Changes in requirements after project has begun

The requirement defects should be prevented otherwise the flow of design and the source code faces a downstream. Since requirements are the most important aspect of producing quality software, the Q-Use Case proposed in this study leads to solve many laborious problems and errors that might occur during elicitation technique [18]. By improving requirements elicitation, the requirements engineering process can be improved, resulting in enhanced system requirements and potentially a much better system.

3.1 Need for Q-Use Case

We accept that testing the software is an integral part of building a system. However, if the software is based on inaccurate requirements then, despite well-written code, the software will be unsatisfactory. Instead of limiting our testing to code, we should start testing as soon as we start work on the requirements for a product [6]. The aim of this work is to overcome the defects related to use case as early as they can be identified and hence prevent them from being reflected in the design and implementation. By incorporating QUC proposed in this study, many of the requirement elicitation defects can be avoided.

3.2 How to uncover requirement defects in Q-Use Cases?

Our Q-Use Cases aims to provide quality use case models which focuses on four factors where each factor ensures that the use cases are correctly framed based on the requirements elicitation, designed, implemented and tested. In the part of testing we check whether the use case developed for software is a quality one or not [6]. In order to check its quality and recover from its defects we introduce "C*heck Condition Testing*". The following four factors need to be considered to recover defects from use case.

- Correctness
- Completeness
- Clarity
- Consistency

4. PROPOSED SYSTEM:-

In comparing with other methods use case reduces complexity and it creates a mutual understanding between the project builders and the reader's .This section organizes the ideas of the authors improving the software requirement elicitation method using Q-Use Case.

The Q-Use Case will perform *Check Condition testing* using the following attributes:

Correctness: Every requirement should describe each function that has been used in the system. Only when the function is described the cycle will be understood by the reader. The user can only inspect the correctness of the system. If the system fails to describe its function, there are chances of the failure of the system. The use case should contain all that is required to answer the problem

Completeness: The limitations and features of the function and its work of nature in its environment must be known before it is implemented in the system. The set of uses cases representing the function of the system should be complete i.e. every function should act on its own and along with other

functions to achieve the goal. There should not be any or misleading information during elicitation process. The use case must lead a logical path with events in the description in the correct order.

Clarity: The phrases describing the use cases and actors should meet the requirements of the user i.e. it should be understandable and must give a clear view of the system to the user. Simple present tense should be used throughout the description. Adverbs, adjectives, pronouns, synonyms and negatives can be avoided while framing the description.

Consistency: Each requirement should document something the customers really need or something that is required for conformance to an external requirement. If all the requirements are regarded as equally important, the project manager is less able to react to new requirements added during development, budget cuts, schedule overruns, or the departure of a team member. Common terms should be defined and used across all use cases. It should link each software requirement to its source that is a higher-level system requirement, a use case, or the customer statement. Figure 2 demonstrates the process of check condition testing by the quality personnel. The quality personnel use the Q-Use Case to check for the correctness, completeness, clarity and consistency.

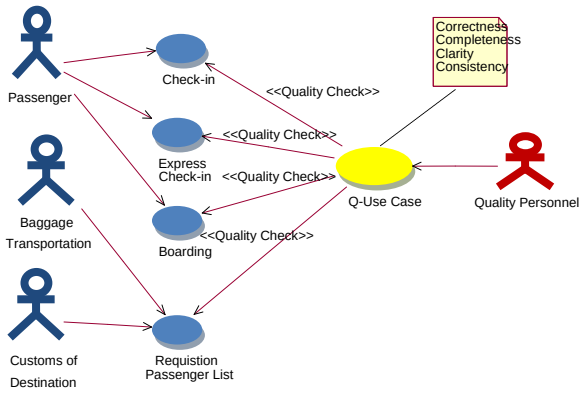

Figure 2 Integration of Q-Use case in use case modeling

4.1 Q-Use Case Scenario

Q-Use Case checks the use cases in the system in quality aspect. Once the use cases and its description are described in the use case model, the quality personnel (one who performs quality audit) should check the use cases involved in the system in terms of 4C 's Correctness, Completeness, Clarity and Consistency in order to overcome the anomalies in the requirement elicitation process. The steps taken by quality personnel in doing the *Check Condition testing* involves the following:

Step 1: Check for Correctness:-

i. *The quality personnel will check for the functioning of the other use cases using the Q-Use Case.*

ii. *It checks whether that every use case precisely state one or more functional requirements used by the system.*

iii. *It also checks whether every use case used in the system are prioritized.*

iv. *The QUC identifies the malfunctioning of a use case.*

Step 2: Check for Completeness:-

i. *The QUC will check that the use cases used in the system interprets all of the functional requirements.*

ii. *The QUC checks whether that all use case used in the system is necessary and have a role to be played in the system.*

iii. *The QUC detects for the misleading path in the system.*

Step 3: Check for Consistency:-

i. *The QUC checks whether that every use case goes hand-in–hand with the co-coordinating use cases.*

ii. *It checks whether the use cases are non- conflicting.*

iii. *It checks if there is any redundant information in use case scenario.*

iv. *It checks whether common terms like specific events or actors are defined consistently across all use cases.*

v. *It checks whether all use cases are externally consistent with other modeling events like class diagram, object diagram etc.*

Step 4: Check for Clarity:-

i. *The QUC checks for Simplicity of the use case.*

ii. *The QUC checks whether the use cases used in the system are sufficiently precise and unambiguous (i:e) It checks whether the use cases are understandable to those who will need to work them later.*

iii. *The QUC also checks whether the use cases have only one interpretation.*

iv. *It also checks if the terms used for specifying actors are included in the dictionary and glossary for the purpose of clarity to other stakeholders in the system.*

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. Activity diagrams seizure the dynamic behavior of the system and does not have a message flow from one activity diagram to another. Activity is the main element of the diagram itself. It is the function

carried out by the system. The activity diagram should identify these four elements:

- Activities
- Association
- Conditions
- Constraints

This activity diagram depicted in fig 3 performs the following activities:

- Send use case description for verification by Quality Personnel(QP)
- QP confirms the receipt of use case.
- QP Checks the use case for 4C's using check condition testing
- If the testing is satisfied, use cases passes the condition testing
- If it is not, the use case scenario after further interaction with actor.

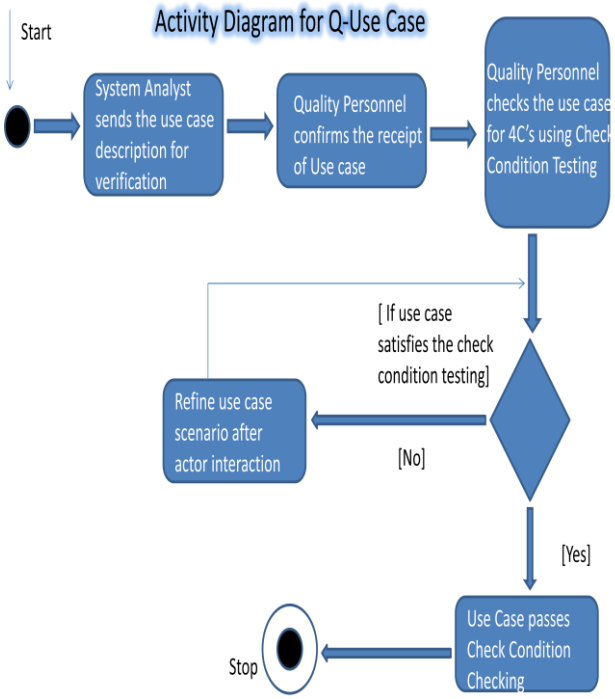

Figure 3 Activity Diagram for Q-Use Case.

5. BENEFITS OF USING PROPOSED SYSTEM:-

Using Q-Use Case, the functions proposed in the system are checked for their quality**.** Q-Use Case crystallizes your vision of the product's behavior as specified in the requirements and can reveal the omissions and ambiguities in the system.

It does not make the reader to guess the work or functioning of the system. It provides reality check on what can be done and cannot be done technically, and what can be done only at excessive cost .It allows you to trace each requirement back to its origin. It adds priorities to requirements where the project manager will be able to react to requirements that were newly added during development. The reader exactly gets one clear picture of the interpretation. It checks whether each requirement is simple and straight forward and does not include technical jargons. The model can be lead to more inspections and demonstrations where it is doubly verified. It is hard to spot missing requirements because they aren't there. Consistent requirements do not conflict with other software requirements. Requirements go hand in hand with each other. The model can be tested for quality even after some modification applied to the system. Each requirement can be referred easily as they are uniquely labeled. It easily identifies the errors or the things which are left out while designing.

6. CONCLUSION

Use case diagram exhibits the interplay between actors and users. Requirement elicitation techniques are the most competing technique in software development. In this paper, we have described about the effectiveness of using Q-Use Case in requirement elicitation to improve software development. This paper also highlights the importance of 4C's - Correctness, Completeness, Clarity, and Consistency in use case description. The actor (quality personnel) in the use case diagram (fig 2) plays a vital role in adopting Check Condition Testing for normal use cases in the use case diagram.

In this study, the role played by the quality personnel when checking the 4 C's is mentioned in the form of Q-Use Case scenarios. The Q-Use Case used in this study help us to realize its importance in achieving quality in requirement elicitation

The Check Condition Testing that we have proposed in this paper limits its verification of use case with respect to 4C's. This study can be extended by adopting more quality check parameters in Check Condition Testing thereby we can prevent most of the defects occurring in requirements gathering which in turn would help us "to build the right product".

7. REFERENCES

[1] Improving Requirements Quality using Essential Use Case Interaction Patterns, Massila Kamalrudin, John Hosking, John Grundy *ICSE'11*, May 21–28, 2011, Honolulu, Hawaii, USA.

[2] Eliciting security requirements with misuse cases Guttorm Sindre Æ Andreas L. Opdahl Received: 15 February 2002 / Accepted: 5 March 2004 / Published online: 24 June 2004 _ Springer-Verlag London Limited 2004

[3] Requirements Elicitation with Use Cases Shane Sendall and Alfred Strohmeier Swiss Federal Institute of Technology in Lausanne Software Engineering Lab

[4] An MKS White Paper By Dennis Elenburg Application Engineer

[5] 7 "S" of Defects Occurrence – A Case Study, Arupratan Santra

[6] Describing Use Cases with Activity Charts, Jes´us M. Almendros-Jim´enez and Luis Iribarne

[7] Analyzing User Requirements by Use Cases: A Goal-Driven Approach, Jonathan Lee and Nien-Lin Xue, National Central University

[8] Analysis of use case approaches to requirements engineering, Virpi Mäkinen

[10] Bjorn Regnell, Requirements Engineering With Use Cases – a basis for software development , Department of Communication Systems, LUND University, Lund 1999

[11] Bagiampou, M. A Use Case Diagrams ontology that can be used as common reference for Software Engineering education , Fac. of Math., Univ. of Patras, Patras, Greece Kameas, A. ,6-8 Sept2012

[12] Ruth Malan and Dana Bredemeyer, Functional Requirements and Use Cases

[13] Chaelynne M. Wolak, Gathering Requirements The Use Case Approach, School of Computer and Information Sciences , Nova Southeastern University ,June 2001

[14] Andrew Gemino, Drew Parker, Use Case Diagrams in Support of Use Case Modeling: Deriving Understanding from the Picture, Simon Fraser University, Canada

[15] Søren Lauesen & Otto Vinter, Preventing Requirement Defects, IT University