Confusions in Writing Use Cases

Ann Johansson
Department of Informatics and Mathematics
University of Trollhättan/Uddevalla
ann.johansson@htu.se

Abstract

Use cases are often very powerful and are popular to use when defining functional requirements for a system. UML supports the use of use cases in object-oriented systems development. However it is not always clear for systems developers on how to use use cases. It can be very confusing in knowing what to include or how to structure use cases. In this study a weather station system has been analysed with an object-oriented approach. Some problems occurred on how to structure scenarios and use cases. Problems also arose on what to describe in the use cases. The problems were analysed and assessed in this paper. The paper points out that the need for guidelines is of great importance.

1. Introduction

The use case concept is widely used in object-oriented analysis. And it is also supported by UML. The simplicity of use cases makes them very powerful. Jacobson introduced use cases in 1986 and systems developers immediately found them attractive because use cases imply the ways in which the user uses a system. The functionality of a system is defined of a set of use cases, where each use case represents a special sequence of interaction [5]. Use cases describes functional requirements according to Larman [1]. But people do not find use cases so easy to use. It can be very confusing in knowing what to include or how to structure them. It is hard to decide if an interaction consists of one or more use cases. Cockburn [8] has found over 18 different definitions of the use case concept, which differ along the dimension of purpose, contents, plurality and structure. If is quite apparent that this confusion can lead to poorly-designed systems. As Korson [10] claims that he experienced that use cases more often are misused than used correctly the consequences can be severe. The motivation for use case creation is however to gain an understanding of the problem and a proposed solution and also to identify candidate classes in the conceptual class diagram. But creating use cases is by no means a foolproof process according to Gottesdiener [11].

2. Research method

The study is done within the framework of a systems development project. The goal for the project was to develop a weather station system. Object-oriented method was used for this project and it was documented by UML. The systems development group had a few objectives from the employer to fulfil about the system.

The research is performed as a case study. In the project it was faced problems in writing use cases in the analysis phase. There were for example confusions about defining scenarios. In order to solve the problems in the confusion of use cases and to assess the results a literature study was done.

3. Use cases

3.1 Defining the concept

Some various authors describe the use case concept. Ivar Jacobson introduced the idea of use cases in 1986. Jacobson, Ericsson and Jacobson [9] in 1995 defined use cases as "*a use case is a sequence of transactions in a system whose task is to yield a measureable value for an individual actor of the system*". Larman [1] describes use cases and their use in a detailed way. He says that use cases are requirements, primary functional requirements that indicate what the system will do. Customers and end users have goal and they

want computer systems to fulfil them. The use case view is a static model of the requirements as seen by its end users, analysts and testers [7]. Use cases describe desired behavior, but they do not dictate how that behavior will be carried out [4]. "*A use case is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result to an actor*." [4].

Quatrani [6] states that uses cases model a dialogue between an actor and the system. Actors can then be people or computer systems. She also joins Jacobson et al's definition [9]. Quatrani's rule of thumb is: " *A use case typically represents a major piece of functionality that is complete from beginning to end. A use case must deliver something of value to an actor*." She also recommends a brief description of a use case in a few sentences. Then each use case is documented with a flow of events of what the system should do.

3.2 Scenarios

A scenario is a specific sequence of actions and interactions between actors and the system. It is also called a use case instance. Larman [1] points out that a use case is a collection of related success and failure scenarios. Cockburn [12] has the same view of a scenario. The scenarios describe actors using a system to support a goal. Booch, Rumbaugh and Jacobson [4] also states that a use case describes a set of sequences. Each sequence represents the interaction of the things outside the system with the system itself. If a function might have many possible variations the use case describes a set of sequences. Each sequence is called a scenario. As Booch, Rumbaugh and Jacobson claim, scenarios are to use cases as instances are to classes. It means that a scenario is basically one instance of a use case.

Cockburn [8] discusses about plurality. The question is if a use case contains more than one use case. May a use case really be just another name for a scenario? [12]. A scenario, a sequence of interactions, has no branching or alternatives. Cockburn [12] states that a use case is a collection of possible scenarios between the system under discussion and external actors, showing how the primary actor's goal might be delivered or might fail. The scenarios are separated according to the conditions encountered, and grouped together as they have the same goal.

As Cockburn [12] suggests the characteristic information for a use case is:
1. Primary Actor or actors
2. Goal
3. Scenarios used

The characteristic information for a scenario is:
1. Primary actor
2. Goal
3. Conditions under which scenarios occurs
4. Scenario result (goal delivery or failure)

3.3 Relationships between use cases

There can be different types of relations between use cases. According to Pilone [2] use cases can be related using generalisation, extension or inclusion. Use case generalisation behaves exactly like class generalisation, where the specialised use case inherits the behavior from the generalised use case. An included use case is not used by itself, in can be used only in a part of a larger, separate use case. Use case extensions is used to encapsulate a distinct flow of events that are not considered part of the normal or basic flow. Booch, Rumbaugh and Jacobson [4] also describe generalisation, extension and inclusion. To use theses three types of relations too often can however be resulted in that the use case diagram can be complicated to understand [5].

A rather different way to organise use case is presented by Si Alhir [3]. He suggests that use case should be organised hierachically. Then the use cases are refined into a set of smaller use case. The refining use cases are subordinate to the use cases of the whole.

3.4 Contents in use cases

According to Larman [1] use cases are text documents and they can be written in different formats, black-box versus white-box visibility type and in varying degrees of formality; brief, casual and fully dressed. Black-box use cases specify what the system have to do (the functional requirements) and the "how" decision should be concerned in the design. Fully dressed use cases are more detailed and are structured. Berard [13] have found that it is hard to have an adequate sense of the proper level of detail in use cases.

White-box type of use cases is used in the design. They show how the use cases really can be used in the collaboration between objects and classes [5].

According to Gottesdiener [11] the nonfunctional requirements and GUI constructs should be kept out from the use case text. Cockburn [12] also claims that it is most useful to stay away from the dialog interface during requirements gathering. It is both time-consuming and subject to change when the final user interface is designed.

4. The weather station project

A system to get different information from a weather station was analysed and designed. The system should

be able to fetch, process, store and display current weather data from the weather station as well as historical data from a database. The system should also display a weather prediction based on a comparison of the current weather and stored weather data. A picture from a web camera should also display current weather. A special group of users should be able to compute some weather statistics from the weather system.

Trying to find use cases and actors for the weather system started the project. The use cases and the actors can be seen in figure 1.

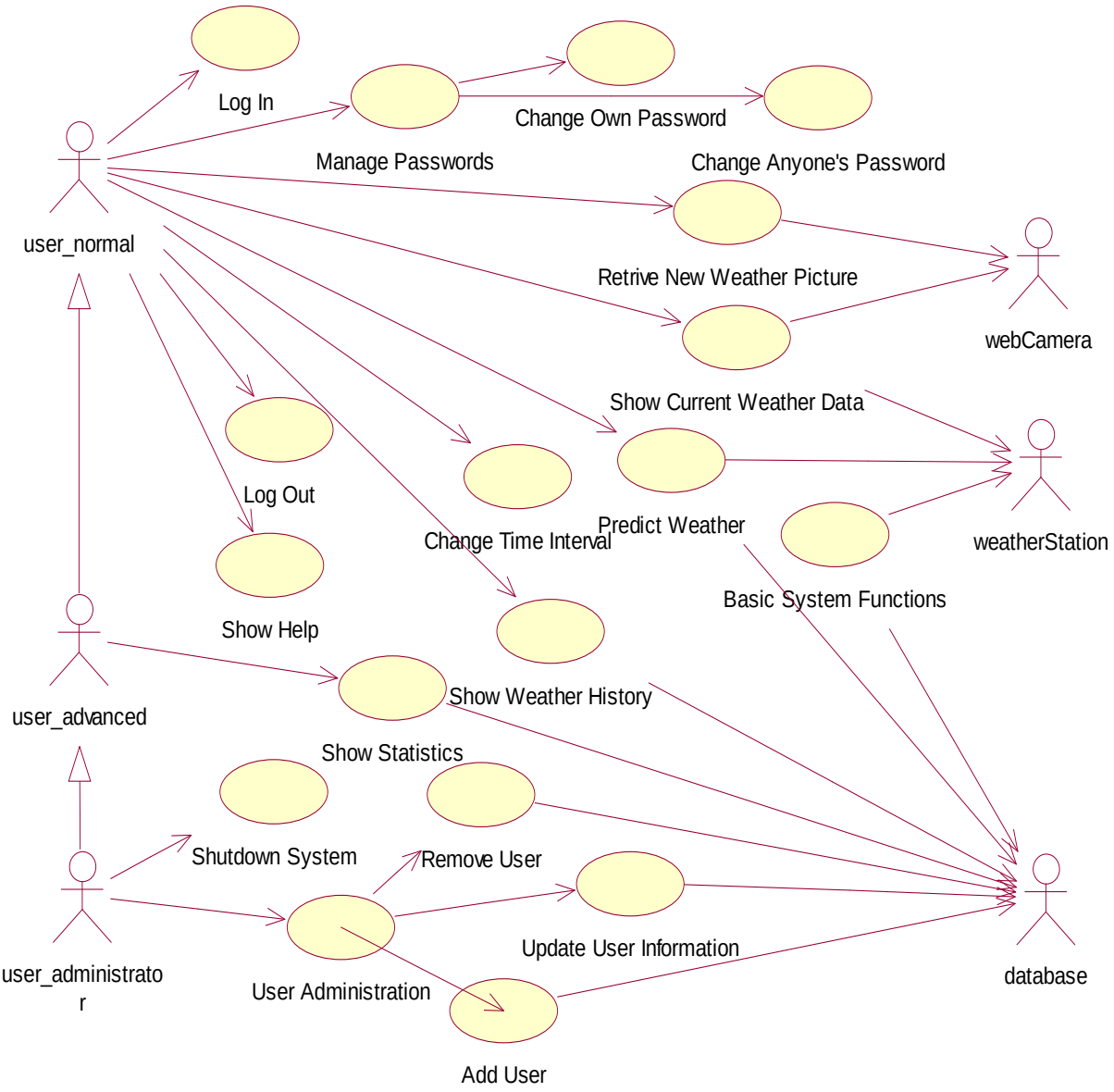

Figure 1 The use case diagram for the weather station system

5. The assessment of results

The writing of use cases started with a high-level description and then greater detailed use cases are described iteratively. This effective way to write use cases was recommended by Gottesdiener [11]. Most of the use cases were completed to fully dressed after the iterations [1]. The striving has been to structure and describe the use cases in detail.

5.1 Assessment of relationships and scenarios

The use case "Manage passwords" is used to illustrate problems in relationships and scenarios in writing use cases (figure 2).

Use case: Manage passwords

Primary actors: User, administrator
Interests: User, administrator

Brief Description
The user or administrator request change of password. The system displays the correct dialog depending on the group membership (admin or not)

Flow of Events

Basic Flow

1. The user request change of password

Alternative Flows

1a. If it is the administrator requesting change of password, *see use case Change everybody's password*.
1b. If it is not the administrator requesting change of password, see use case *Change own password*.

Pre-conditions
User/administrator logged in.

Post-conditions
A successful change of password.

Figure 2 Use case: Manage passwords

As a use case has only one main success scenario or basic flow the use case "Manage passwords", has references to two other use cases (see figure 2, alternative flows) [1]. The user goal is however not fulfilled by the use case "Manage passwords". It has to be continued in one of the two use cases, which are related and referenced in the alternative flows. This use case is not however in accordance with Quatrani's statement that the functionality has to constitute "*a functionality that is complete from beginning to end*" [6]. According to Quatrani should the use case "Manage passwords" have one main flow, with references to subflows in the very same use case. But Pilone [2] writes about use case inclusion. In the exemplified use case "Manage passwords" and the related use cases "Change anybody's password" and "Change own password" can the relation be regarded as use case inclusion. The included use case is not used by itself. The containing use case will be stated in the flow of events when it is invoked. May the solution of the use case "Manage passwords" can be considered as a successful solution as the included use cases never would happen outside of the context of the larger goal. On the contrary the larger goal can never happen and be completed without any of the included use cases either. In this point of view the solution may be less successful. According to Booch, Rumbaugh and Jacobson [4] a scenario is an instance of a use case. The included use cases are then scenarios, or instances of the use case "Manage passwords". To address Cockburns [12] statement the use case "Manage passwords" could have two scenarios; the "change anybody's password" and the "change own password".

5.2 Assessment of contents in the use cases

The use case "Show weather statistics" is used to illustrate problems in deciding interaction details and level of contents in the use cases (figure 3).

Use case: Show weather statistics

Primary actor: Advanced user
Interests: Advanced user

Brief Description
The user requests weather statistics. The system collects the weather data from the database and presents the result on the screen.

Flow of Events

Basic Flow

1. The user requests statistics of the weather.
2. The system displays the "Statistics" dialog.
3. The user chooses between statistics for the last day, the last week or the last month.
4. The system collects weather data for the specified time interval from the database.
5. The system presents the data on the screen.

Alternative Flows

4a-5. The database is unavailable:
1. The system displays an error message.

Pre-conditions
Advanced user logged in.
The user is member of the advanced user group

Post-conditions
The statistics asked for were presented on the screen.

Figure 3 Use case: Show weather statistics

Interaction detail has to do about what is going to be described or not in the use case, in which level interaction will be described. The semantic interface level is chosen in the use case "Show weather statistics"[12]. This level will capture the actor's intention. It is only described that the system displays the dialog for "Statistics". Then the user can choose between statistics for the last day, the last week or the last month. None of the events say anything about how the user interface is designed. There is not described how the user can choose, from a pull-down list how the user can choose among the alternatives in other ways. Ambler [14] also claims that use cases should not describe what the user interface looks like or how it works.

Use cases often need to be more elaborate than they are in the brief format, where the use case is described in a one-paragraph summary [1]. In the use case "Weather statistics" the use case is described in more detail than in the casual format. The casual format describes the use case in informal paragraphs and contains multiple paragraphs that cover various scenarios. The use case "Show weather statistics" is more elaborated, as fully dressed format. All steps and variations are written. However they may have been written in more detail. The Basic Flow is the same as the Main Success Scenario. Still the Special Requirements, Technology and Data Variations List, Frequency of Occurrence and Open Issues are not taken into account, as described by Larman [1].

6. Conclusions

This paper points out that it is not too easy to distinguish between the concept of use case and the concept of scenario. Need of guidelines is of great importance, especially during the first system development projects. Iterations are also very important to complete the use cases in a useful way. The use cases have to be described in detail in a fully dressed format.

7. References

[1] Larman, C, 2001, Applying UML and Patterns, An Introduction to Object-Oriented Analysis and Design and the Unified Process, Prentice Hall

[2] Pilone, D, 2003, UML, Pocket Reference, O'Reilly

[3] Si Alhir, S, 1998, UML in a Nutshell, O'Reilly

[4] Booch, G, Rumbaugh, J, Jacobson, I, 1998, Pearson Education

[5] Kruchten, P, 2002, Rational Unified Process, An Introduction, Addison-Wesley

[6] Quatrani, T, 1999, Visual Modeling with Rational Rose 2000 and UML, Addison-Wesley

[7] Royce, W, 1998, Software Project Management, A Unified Framework, Pearson Education

[8]Cockburn. A, 2001, Writing Effective Use Cases, Addison-Wesley

[9] Jacobson, I, Ericsson, M, Jacobson, A, 1995, The Object Advantage: Business Engineering With Object Technology, Addison-Wesley

[10] Korson. T, The Misuse of Use Cases (Managing Requirements), http://www.korson-mcgregor.com/publications/korson/Korson9803om.htm, 2003-10-17

[11] Gottesdiener, E, Top Ten Ways Project Teams Misuse Use Cases – and How to Correct Them, http://www.thereationaledge.com/content/jul_02/t_misuseUseCases2_eg.jsp, 2003-10-17

[12] Cockburn, A 1997, Structuring Use Cases with Goals, Journal of Object-Oriented Programming, Sep-Oct 1997 and Nov-Dec 1997

[13] Berard, e, V, Be Careful With "Use Cases", http://www.tota.com/pub/use_cases.htm, 2003-10-17

[14] Ambler, S, W, Use Case Modeling Tips, http://www-106.ibm.com/developerworks/java/library/ws-tip-uml-2.html, 2003-10-17