

USE CASE MODELING FOR REQUIREMENT SPECIFICATIONS

Dr. Ajay D. Shinde

Associate Professor

Department of Computer Studies

Chhatrapati Shahu Institute of Business Education and Research, Kolhapur, Maharashtra, India.

Abstract: Requirement engineering is foremost and essential activity in software development process. Its impotence is evident, model building is time-consuming as it emphasizes the understanding of what is required by the user and what should be given to the user. The model built using use case is a combination of diagram and text, the only diagram is not sufficient to specify user requirement. To support diagram some textual information is must, this makes model self-explanatory. This paper presents the idea about requirement specifications using use case model, where proper combination of diagram and text will be presented to make the idea clear.

Keywords: Use case model, requirement specifications, object-oriented analysis, actors, requirement engineering.

I. INTRODUCTION.

In software industry, various approaches are used for specifying user requirements. Building the use-case diagram is a crucial task as it represents user requirements that need to be mapped on to design model during design phases. However, building use case model may consume lots of time and efforts; in addition to this, it requires the complete understanding of the requirements [4].

The use case diagram was first proposed by Ivar Jacobson in 1986 [8]. A use case modeling has become a standard object-oriented methodology used in system analysis to identify, clarify, and organize system requirements. Use case diagrams are part of UML (Unified Modeling Language) that is accepted as a standard notation for the modeling of real-world objects and systems [7]. In the Unified Modeling Language (UML), models used to represent system are classified into two categories structural and behavioral; a use case diagram is a part of the behavioral model [5]. The use case model helps the developer to capture functional requirements; also it serves as documentation of user requirements. It is used to define the functional scope of the system [1].

While using UML-based design and development approach, the first step is to identify use cases and is used to form use case diagrams. After this, each use case is described in detail through textual descriptions. This, results in a composite model having two parts. The first part is a UML model, which shows the use cases and their relationships, the

second part is a set of textual description that define the behavior of use cases shown in the use case model. Although these two parts represent different views still they complement each other very well [9]. In rest of the sections, we will discuss different concepts that are part of use case model and their importance in requirement specifications.

In [2], Hans-Erik Eriksson et al. have given a clear-cut method for use case modeling. It includes following steps.

- Identify actors.
- Find out the relationship between them.
- Identify use cases.
- Find out the relationship between them.
- Draw use case diagram.
- Write use case descriptions.
- Verify and validate your use case model.

II. USE CASE MODEL

As discussed section I of this paper, use case model is having two parts, the first part is diagram called as use case diagram. The use case diagram is a diagrammatic representation of user requirements.

A. Use Case Diagram Concepts

- Actors - external users of the system and user may be human being playing different

roles in the system or may be any other system that is interacting with the system.

- Use cases – functionalities expected by actors from the system. Here functionality is the work to be performed by the system.
- Association – the relationship between actor and use cases.

Here, the meaning of an association is an actor participates in the behavior described by these case. This means an actor is involved in interaction with the use case or interested in result generated by use case [3].

B. Notations used to draw diagram

The following table shows notations used to draw a use case diagram.





Notation	Name	Meaning
	Use case	Work to be done by system
	Actor	Somebody or something interested in the system.
	Association	link between actor and use case
	Relationship	Relationship between use cases and actors

Table 1: Notations used to draw use case diagram

C. Relationship between use cases and actors

The use cases shown in use case diagram may have relationship between them; broadly these relationships are classified as

- Includes – it implies that one use case is an integral part of another use case. Every time the use case invokes a call to another use case.

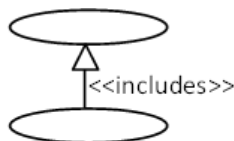


Figure 1: Includes relationship between use cases

- Extends – it implies that one use case may call another use case if required. Here

whether to call another use case or not will be dependent on certain condition.

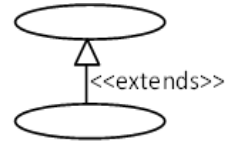


Figure 2: Extends relationship between use cases

The use case diagram may also show the relationship between actors, there is only one type of relationship possible between actors and that is the generalization.

D. Use case description.

Only use case diagram is not self-sufficient for requirement specifications. It must be supported by the appropriate textual description in order to make it complete. In [6] Peter Haumer, calls use case description as use case specifications, that not only defines functional requirements but also describes input-output scenarios with possible variations and expectations. Different researchers are proposing various formats for specifying use case description, but essentially all formats record certain common things such as flow of messages, an alternative flow of messages etc.. Whatever format you choose the use case description should provide a complete specification for use case under consideration.

III. USE OF USE CASE MODEL: AN EXAMPLE

A. Identifying and modeling actors

Let us consider an example of college library system, where the main actors are borrowers and library staff. Where borrowers are further classified as staff and students, in turn, the staff is classified as teaching and non-teaching. Similarly, library staff is classified as librarian, clerk, and attendant. The roles borrower and library staff are general roles and teaching, non-teaching, student, librarian, clerk, and attendant are more specific roles. The relationship between actors playing different roles is modeled as follows.

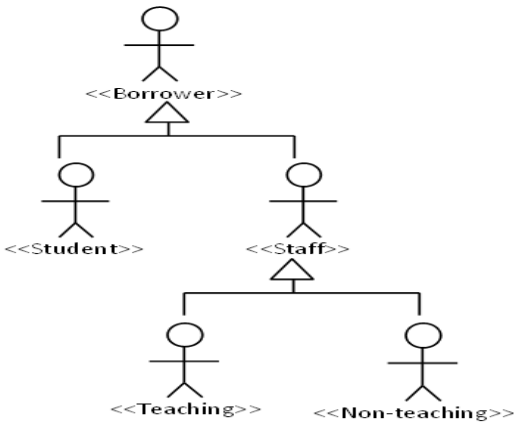


Figure 3: Relationship between actors

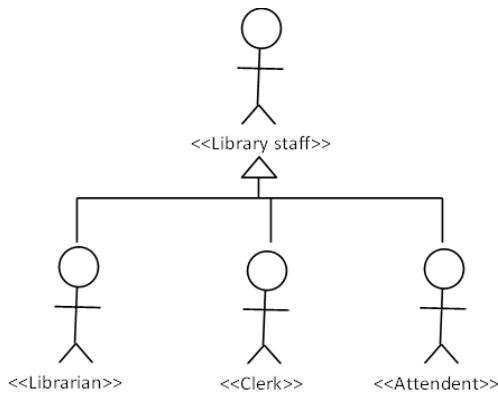


Figure 4: Relationship between actors

The only relationship possible between users is the generalization, which shows the relationship between more general and more specific roles [2]. This will help the developer to understand requirements of the actors while playing these roles.

B. Identifying Use cases and relationship

Once actors are identified, select each actor one by one look and find out the functionality that they may expect from the system. This may produce a big list of functionalities; remove redundancy from the list, now what is left behind is the final list of functionalities expected by actors of the system. Here, each functionality represents a use case, which further may be divided in to use cases that may become part of another use case. This is accordance with modularity principle of design. For college library system the list of functionalities is

- Register borrower
- Register book.
- Place book demand.
- Issue a book.
- Return a book.
- Calculate fine.
- Print fine receipt.
- Print daily issue-return register.
- Print borrower register.
- Print fine collection report.

The functionalities identified above are use cases in the proposed system, the use cases identified above may be further divided into modular use cases e.g. issue a book use case may be further divided in to search borrower, search book, and record issue transaction. This relationship is represented as.

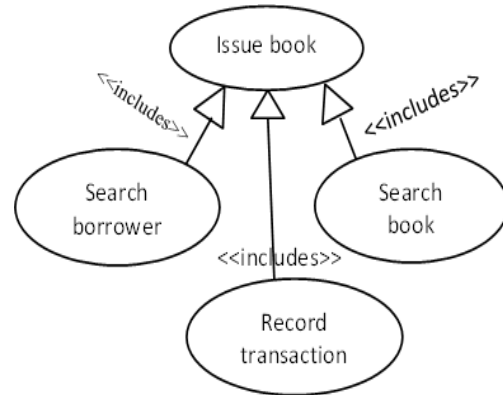


Figure 5 : Relationship between use cases

The includes relationship between use case indicates that every time when the issue book use case is executed search borrower, search book and record transaction use cases must be implicitly executed.

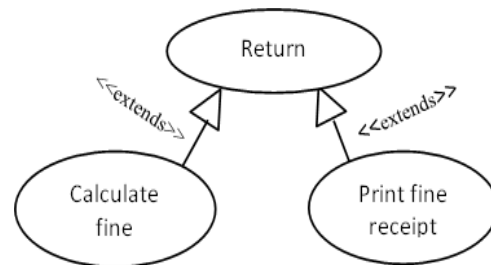


Figure 6 : Relationship between use cases

The extends relationship indicates that every time return book use case is executed, not necessarily calculate fine and print fine receipt will be executed. The execution of calculating fine and print fine receipt is dependent on the probable date of return and actual date of the return for the book. If actual date of return is greater than the probable date of return then and then only calculate fine use case is executed.

Once the relationship between actors and use cases is identified use case diagram is drawn. The use case diagram shows an association between actors and use cases. Where we can understand the actors that are involved/interested/initiate execution of use case.

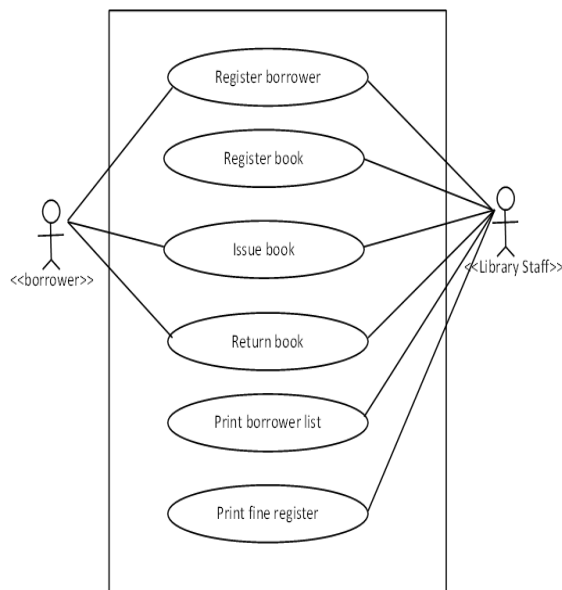


Figure 7: Use case diagram for college library

Figure 7 shows use case diagram for college library system, where big rectangle that includes use cases show system boundaries and actors are external users outside the system boundaries.

C. Writing Use case description

Only use case diagram is not sufficient for describing the complete system and its functionalities. It should be complemented by a little textual description that makes the use model complete and clear. To write use case description, in [2] a template is given, this template helps us to make use case model complete. The given template is

- i. **An objective for the use case:** This describes the main objectives of use case.
- ii. **How the use case is initiated:** It documents which user starts execution of use case.
- iii. **The flow of messages between actors and the use case:** This part documents interaction between actor and use case if everything is normal.
- iv. **Alternate flow in the use case:** This section documents error handling and exception handling by use case.
- v. **Special/supplementary requirements applicable to the use case:** This section documents data definitions, performance requirements of specific steps, security requirements, and so on.
- vi. **How the use case finishes and its value to the actor:** This section documents final results obtained from the use case.

The use case description has to be written for each and every use case shown in use case diagram.

For college library system let us write use case description for issue a book use case.

Name of Use case: issue a book

- i. **Objectives :**
 - To issue required book to the borrower.
 - To record transaction permanently.
- ii. **Initiator :**
 - Borrower initiates the issue use case
- iii. **The flow of messages :**
 - The system prompts a message enter borrower number.
 - Library staff enters borrower number.
 - The system displays borrower's information after search.
 - System prompts a message enter book number.
 - Library staff enters book number.
 - The system displays book information after the search.

- Library staff selects the action to record issue transaction permanently.
- iv. An alternative flow of messages:**
- If borrower number is invalid system prompts a message ‘Invalid borrower number’
 - If book number is invalid system prompts a message ‘Invalid book number’
 - If book is already issued to borrower system prompts a message ‘Book not available’
- v. Special/supplementary requirements:**
- The barcode reader is required for reading borrower number and book number directly from borrower’s card and book.
- vi. Result**
- Use case finishes by recording issue transaction information permanently in issue book, the book required by the borrower is handed over.

The description written above will make use case issue a book completely. If the description is written for each use case shown in use case diagram it will provide developer and tester with everything that they need.

IV. CONCLUSIONS

Requirement engineering forms the base for every software development process. For modeling requirements, different techniques are used and every technique has some advantages and disadvantages. The use case modeling is a technique for requirement specification under unified modeling language. This technique helps the developer model user requirements in an appropriate manner. It not only provides a way for representing user requirements diagrammatically but also complements it with some textual description to make use case model complete and clear. The use case model is easy to use and understands and will help developers and testers understand user requirements in a better manner.

REFERENCES

- [1] BENTE ANDA, DAG I. K. SJØBERG, *Investigating the Role of Use Cases in the Construction of Class Diagrams*, Springer Science + Business Media, Inc., pp 285–309,
- [2] Hans-Erik Eriksson, Magnus Penker, Brian Layons, David Fado, *UML 2Toolkit*, Wiley Publishing, ISBN: 0-471-46361-2
- [3] Luis Iribarne, *Describing Use-Case Relationships with Sequence Diagrams*, Article in *The Computer Journal* · January 2007, DOI: 10.1093/comjnl/bxl053
- [4] Mohammad I. Muhairat and Rafa E. Al-Qutaish, *An Approach to Derive the Use Case Diagrams from an Event Table*, *Proceedings of the 8th WSEAS Int. Conference on SOFTWARE ENGINEERING, PARALLEL and DISTRIBUTED SYSTEMS*, ISSN: 1790-5117, ISBN: 978-960-474-052-9
- [5] Munassar, N. ; Govardhan, A. (2011) *Comparison between Traditional Approach and Object-Oriented Approach in Software Engineering Development*, *IJACSA, International Journal of Advanced Computer Science and Applications*, Vol. 2, No. 6,
- [6] Peter Haumer, *Use-case based software development*, *IBM Rational Software, Scenarios- Part 2: Techniques*.
- [7] *Use case diagram* <http://whatis.techtarget.com/definition/use-case-diagram>
- [8] *Use case diagram (1)*, available via http://en.wikipedia.org/wiki/Use_case_diagram, Last modified 18 September 2015,
- [9] Veit Hoffmann, Horst Lichter, Alexander Nyßen and Andreas Walter, *Towards the Integration of UML- and textual Use Case Modeling*, *JOURNAL OF OBJECT TECHNOLOGY*, Vol. 8, No. 3, May-June 2009