Guidelines for Developing Quality Use Case Descriptions

Yunan Chen, Drexel University, Philadelphia, PA 19104, USA; E-mail: yunan.chen@drexel.edu
Il-Yeol Song, Drexel University, Philadelphia, PA 19104, USA; E-mail: song@drexel.edu

ABSTRACT

A use case description (UCD) is used to document detailed behavior of a use case in order to communicate its functionalities to different stakeholders related to the use case. A UCD plays an important role throughout software project's lifecycle. But there is no standard or widely-accepted approach for developing UCDs. They are developed based on the personal preference; both UCD formats and contents vary largely among different documenters. In this paper we present the seven-step use case documentation method. Our method integrates two writing rule sets and the three-level hierarchical use case evaluation method. The two writing rule sets cover syntax style and step rules that guide how to write each step in UCDs. The three-level hierarchical use case evaluation method helps use case developers evaluate UCDs from the overview level, the use case element level, and the sentence level. The recommended techniques of our paper is a synthesis of a thorough comparison of various UCD contents and formats discussed in literatures and our own experiences developed through a graduate Systems Analysis and Design class for many years.

1. INTRODUCTION

A use case approach is widely used to model system functionalities. A use case is a collection of success and failure scenarios in achieving a goal of an actor. A use case model consists of a *use case diagram* and a *use case documentation* (UCD). A use case diagram succinctly summarizes system behaviors from the point of view of actors. A UCD describes use case behaviors and functions in a narrative structured text file [1]. The documentation could be supplemented by diagrams such as activity diagrams and sequence diagrams. These diagrams provide a visualized flow of system interactions. The textual document, however, is the most common and understandable approach for UCDs [7, 8, 15].

A UCD is a communication tool which helps different stakeholders to understand the use case and provides supplementary information for system specification. As a communication channel, readability and understandable is the primary goal for a good UCD. Also since UCDs serve for software system specification, there are extra requirements than the regular text documentation. UCDs should clearly and completely depict system requirements for the specification. Narrative descriptions, however, usually tend to be ambiguous and lack a structure. There must be a set of coherent guidelines to direct documenters in content and structure selection.

A UCD plays an important role throughout software project's lifecycle. But there is no standard or widely-accepted approach for developing UCDs. They are developed based on the personal preference; both UCD formats and contents vary largely among different documenters [1-8, 11, 13-16]. In spite of the important roles of UCDs throughout the project's lifecycle, there have been only few guidelines for producing good UCDs. The lack of accepted guidelines makes both writing and assessing UCDs difficult. It is very difficult to ensure the proper information to be conveyed to all related stakeholders.

In this paper, we present the seven-step method for developing high quality UCD for a given use case diagram. Our method integrates *two writing rule sets* and the *three-level hierarchical use case evaluation method*. The two writing rule sets cover syntax style and step rules that guide how to write each step in UCDs. The three-level hierarchical use case evaluation method helps use case developers evaluate UCDs from the overview level, the use case element level, and the sentence level. The recommended techniques of our paper is a synthesis of a thorough comparison of various UCD contents and formats discussed in fifteen literatures and our own experiences developed through a graduate Systems Analysis and Design class for many years.

The rest of the paper is organized as follows. Section 2 first synthesize the previous work on methodologies for developing UCDs and then give a literature review. Section 3 presents the seven-step UCD development method. Section 4 concludes our paper.

2. A REVIEW OF PRACTICES ON UCDS

Our review on UCDs shows that there is no consensus on the well-accepted methodology for writing UCDs. Based on our reviews on literature as well as our own experiences, we believe that there are two aspects in improving the merits of UCDs as a communication and a specification tool: 1) a need for more concrete writing guidelines for UCD documenters; 2) a need for a methodology for assessing the quality of UCDs and removing mistakes.

2.1 UCD Writing Guidelines

A UCD writing process is no easy task since different writing styles may affect the usability and readability of the UCD. The review of the literature results in only a few simple writing guidelines. One notable guideline is the CREWS (Co-operative Requirements Engineering with Scenarios) Use Case Authoring Guidelines [3]. CREWS consists of eight specific rules. It is believed to be the most complete guideline available till now. But even these rules are quite abstract and hard to remember [4]. To further enhance usability, Cox and Phalp simplified the CREWS to a four step guideline called CP rules [5]. A summary of CREWS and CP rules are shown in Table 1. We will discuss how to remedy the limitations of CREWS and CP rules in Section 3.5.2.

2.2 Literature Review on UCD Evaluation Approaches

UCDs need to be assessed before they are used for system design and implementation. Some experts [6, 2, 7] mentioned that precision and clarity are key

Table 1. A summary of CREWS and CP rules [3, 5]

CREWS rules	CP rules
Style 1: Each sentence in the description should be on a new, numbered line. Alternatives and exceptions should be described in a section below the main description and the sentence numbers should agree. Style 2: Avoid pronouns if there is more than one actor. Style 3: No adverbs or adjectives. Style 4: Avoid negatives. Style 5: Give explanations if necessary. Style 6: All verbs are in present tense format. Style 7: There should be logical coherence throughout the description. Style 8: When an action occurs there should be a meaningful response to that action.	Structure 1: Subject verb object. Structure 2: Subject verb object prepositional phrase. Structure 3: Subject passive. Structure 4: Underline other use case names.

factors for assessment. But they are quite subjective and hard to measure. Most researchers provide simple guidelines on how to write a description. But none of them addressed the issue of evaluating the whole UCDs. More comprehensive criteria are needed to solve this issue. Our review found two types of evaluation methods: *Factor approaches* that address important aspects that a good UCD must meet and C*hecklist approaches* that lists questions for assessing a UCD.

2.2.1 Factor Based Evaluation Approaches

Cox and Phalp [5] proposed a method called 4-independent factors - *Plausibility*, *Readability*, *Consistent structure*, and *Alternative flows*. Later, Cox, Phalp and Shepperd [8] came up with a more concrete 4-C use case heuristics which address the communicability of use cases. The concept of 4-C consists of Coverage, Coherence, Consistency, and Consideration of alternatives. Phalp and Vincent further specified the 4-C heuristics into a 7-C approach [4] based on the empirical study they conducted. In the 7-C heuristics, every criterion consists of several sub-criteria. The 7-C heuristics include:

1. **C**overage: the use case should contain required information in relevant details.
2. **C**ogent: the use case has to be complete, logical path follows the correct order.
3. **C**oherent: the written styles should be coherent to be understood.
4. **C**onsistent Abstraction: use case should follow a consistent level of abstraction.
5. **C**onsistent Structure: alternatives should be separated from the basic flow.
6. **C**onsistent Grammar: Using simple present tense and avoid adverbs, adjectives, pronouns, synonyms and negatives.
7. **C**onsideration of Alternatives: alternatives should contain all the possible paths.

2.2.2 Checklist Based Evaluation Approaches

Though the 7-C approach is quite comprehensive, it lacks the requirements of how much details should be included in the UCD. A checklist based approach assesses UCDs from various users' perspectives: customers, designers and testers [9]. Another checklist approach to address the different users' requirements is 3-C checklists proposed by Copeland [10]. The 3-C represents Complete, Correct and Consistent. This approach is to test 3-Cs of a use case from the point of view of syntax, domain, and traceability:

- Syntax testing is to verify that a UCD contains correct and proper information. Copeland claims that more than half of the use cases failed syntax test in a project he conducted.
- Domain expert testing is to check whether the description meets the domain knowledge requirements.
- The traceability testing is to ensure that all the functional requirements are represented in the UCD and can be traced back.

3. THE SEVEN-STEP USE CASE DOCUMENTATION METHOD

Our review showed that there are only few writing guidelines and inspection methods available. It is necessary to propose a comprehensive and understandable method to documenting and assessing UCDs. In this section, we propose our seven-step UCD method which synthesizes existing rule sets and our own rules. Note that our method focuses on UCDs, rather than use cases themselves. Therefore, our method begins from developed use case diagrams.

Table 2. The outline of our seven step guidelines

Step 1	Understand actors and their goals
Step 2	Write use case goals in one phrase
Step 3	Write an overview description in a few sentences
Step 4	Define the precondition and postcondition
Step 5	Write the sequences of interactions
Step 6	Document non-functional requirements and other optional information
Step 7	Evaluate use case descriptions

We propose the following seven-step use case documentation method. It is a set of coherent guidelines covering UCDs from the very beginning to the end. Especially, our focus is on the Writing Rules and Evaluation method of UCDs. Due to the space limit, we only outline the steps without detailed examples. The seven steps of UCD development guideline are summarized in the Table 2. We discuss them in further details below.

3.1 Understand Actors and Their Goals

The first step in writing a UCD is to understand the actors and their goals. Knowing who the actors are and what their goals are help developers write steps of a UCD. Goals of an actor are high level responsibilities of the actor in the system. A goal should represent "what" of a responsibility, not "how" of the responsibility.

3.2 Write a Use Case Goal in One Phrase for Each Use Case

In this step, a developer states the goal of a use case in one phrase using the format of Verb + Noun phrase. This simple phrase will define the specific goal at a high level term to distinguish one use case from another. For example, if we have a use case named "Process Rents" in a video rental system, the goal phrase could be "To capture rental items along with payment."

3.3 Write an Overview Description for Each Use Case

The next step is to write a short summary of the actor-system interaction [11] in a few sentences. The brief description states an overview of what you are trying to achieve and the scope of the use case. At this step, use only business terms without any technology-oriented terms. For example, the overview description of the above use case could be "A store employee checks out rental items for a customer by calculating due dates and correct charges. The use case also includes checking for any overdue items. The store employee accepts payments for the items and any late payments. A rental slip is issued and kept by the store employee."

3.4 Define the Precondition and Postcondition

Preconditions and postconditons could help developers set up the boundary of the use case. These conditions would limit the sequences of interaction into a clearly starting and ending situation [11]. A use case that was performed earlier could affect the preconditions of the others; some use case is included or extended from others. All these relations will affect the content in a UCD. Before start writing the detailed steps, the documenter must understand the use case suite well and know relationships among the use cases so that relationships among use cases are represented and managed correctly, consistently, and completely. A good precondition is the one set by another use case if one needs to be executed after another.

Postconditions are lists of the conditions that must be true after the use case successfully finishes. Larman suggests that postconditions be documented in a passive and past sentence to represent what already happened [15]. Larman also recommends the following three types of postconditions – (a) objects that need to be created/deleted (b) data that need to be changed, and (c) associations that need to be connected/disconnected.

3.5 Write the Sequences of Interactions

The sequences of interaction are the major part of a UCD. It carries the communication and system design function of the UCD. We will give more focus in this step. There are three types of interactions.

3.5.1 Write the Main Successful Scenario

We begin with the main successful scenario first. It represents the most common and successful path. To identify the main successful scenario, we need to start from the triggering event, proceed step by step till the use case reaches the postconditions depicted before.

3.5.2 Write the Other Successful Scenarios

We then write other successful scenarios that refer to alternative scenarios. They are less frequently executed than the main success scenarios, but still achieve the goal of the actor.

3.5.3 Write Unsuccessful Scenarios

We then write other unsuccessful scenarios that refer to those scenarios that stop before the use case goal is achieved

The Two Writing Rule Sets for Developing UCDs

CREWS and CP rules as we reviewed before focus on the syntactic aspect but miss the specific guidelines such as how to develop each step; what information to record in each step; and what information to avoid. To address these issues, we present synthesized rule sets for writing steps, integrating the ideas of CREWS and CP rules as well as our own experience. We present the guidelines in the form of two writing rule sets as shown in Table 3.

Rule set 1: The Syntax Rules - Describe steps in a precise and unambiguous way

- *Use specific nouns*: Avoid using vague terminology like information, data [11]. Specify the data to be created, deleted, changed or associated [1].
- *Avoid pronouns*: If there is more than one actor involved in the UCD, using pronouns will confuse users on which actor this pronoun is referring to [3].
- *No adverbs or adjectives*: A UCD is to depict the goal fulfillment endeavors, not to write a story. Don't use any adverbs or adjectives like appropriate, required, relevant or sufficient [3, 11]. Using these adjectives make the sentence ambiguous.
- *Use straightforward and specific verb*: Avoid using verbs that have overloaded meanings such as get, keep, have, or do. Try to use specific verbs or associate an overloaded verb with an object as in *find a customer name*.
- *Using present tense*: Write in "present tense" to describe what the system does, rather than what it will do or already done [11].
- *Avoid negatives*: Document use case in affirmative way, don't use "not or no" in the description [3].
- *Using active voice*: Using direct and declarative statements started by an actor or the system [11]. For example: document a step as in "the system validates the amount entered" instead of "the amount entered should be validated by the system".
- *Avoid compound sentences*: Using simpler grammar [13], [4, 6] is recommended to adapt "Subject verb object" or "Subject verb objects prepositional phrase" style when documenting the flow of events.

Rule set 2: The Step Rules - Each step should be only one logical step towards the use case goal

- *Each step must be a goal-driven movement*: Describing the user's step at user interface level is one of the most common mistakes in recording use case steps. For example: "system asks for name, user enters name, system ask for address, user enters address" could be replaced by "user enters name and address" [6].
- *Each action should have a system response*: Any action is users' request sent to the system. Thus it should not occur alone, there must be a system response associated to it [3].
- *Each step must represent only one logical step*: State one logical step at a time. Do not combine two different steps that require a system interaction in one step. For example: "Search products and select the item" are two steps and should be documented in two steps because the system needs to display the output before the second action. [16]
- *Describe your steps in a general tone rather a special case*: The description should be general enough for all the possible variations in this step. For example: "Request one-year subscription" is a special case when applying for a subscription; it should be written as "Select the number of years of subscription." [16].
- *Describe in a right logical order of execution*. Steps should be documented in a logically meaningful order of execution to show step-by-step procession. For example, adding a shipping charge during "Process payment" use case is logically in a wrong order [16].
- *Do not include steps stated in the preconditions*. [16].
- *Do not use ambiguous expressions*. For example, the following are ambiguous steps. "Subscriber enters in demographic information" and "Establish subscription data" [16].

Table 3. A summary of the rule sets

Rule 1: The Syntax Rules	Noun	Use specific nouns
	Verb	Use specific verb; Use present tense; Avoid negatives
	Adverbs	No adverbs
	Adjectives	No adjectives
	Pronouns	Avoid pronouns
	Sentences	Use simple sentences; Use active voice
Rule 2: The Step Rules	Each step must be a goal-driven movement	
	Each step must represent only one logical step	
	Each action should have a system response	
	Describe steps in general tone rather a special case	

3.6 Document Non-Functional Requirements and Other Optional Information

For future references and improving understandability, non-functional requirements can be appended at the end of a UCD. Those non-functional requirements include business rules, performance requirements (response time and throughput), reliability requirements, usability requirements, security requirements, volume and storage requirements, configuration, compatibility requirements, backup and recovery, and any training requirements.

3.7 Evaluation of UCDs

A UCD must be evaluated before actual releasing. The evaluation team should include all the possible stakeholders such as users and designers. We believe that the UCD evaluation process should be an iterative process in which each iteration evaluates different aspects. In this section, we present *the three-level hierarchical use case evaluation method – overall level, use case element level, and sentence level.* Evaluation should begin from the overall level to the use case element level and then to the sentence level.

3.7.1. Check from the Overall Level

This level is to assess the overall structure of a UCD. Issues judged at this level are whether the UCD contains an appropriate level of details and the structure of the use case templates are appropriate. A good UCD should convey all the required information but with no redundancy. There are two major factors that affect the levels of detail:

- Stakeholders' concerns: check if the UCD meets concerns of stakeholders such as end users, developers, and the testers [9].
- Different viewpoints: Depending on whether we adopt an external (black box) or internal (white box) view will affect how much details we need to add to the document and what kind of use case formats we select.

3.7.2. Check from the Use Case Elements Level

The next heuristic goes down to use case element level. It is to test whether the elements included in the UCD are content-wise appropriate and structurally sound. The testing could be conducted using the following 3-C rules [10]:

- Cogent: check the logical paths of the UCD and determine whether it follows a logically correct way.
- Complete: check whether the UCD provides a solution to the problem and check whether the entire possible alternatives are recorded.
- Consistent: check whether the UCD follow the same level of abstraction. The numbering in the main flow and alternatives should also be consistent.

3.7.3. Check from the Sentence Level

After checking the correctness of the use case elements, the next level heuristic goes down to the sentence level. Are the descriptions clear enough for users' to

Table 4. The three hierarchical heuristics of UCDs

Overall	Does the UCD satisfy stakeholders' concerns?
	Is an external or internal viewpoint applied?
Elements	Cogent: Are the logic paths correct?
	Complete: Are all alternative paths included?
	Consistent: Are the levels of abstraction and numbering consistent?
Sentence	Check the syntax rules
	Check the step rules

read and understand? Does each sentence make sense to the readers? The rules we proposed in Section 3.5 could be applied here for evaluation.

The three hierarchical heuristics provide us with a more structured approach which allows users to assess the UCDs from general to specific perspectives. The evaluation from a higher level iteration to a specific level gives assessors priorities in the evaluation process. This could potentially improve the evaluation results and improve the usability of the heuristics.

4. CONCLUSION

In this paper, we have presented guidelines for developing quality UCDs. We have presented the seven step method for writing UCDs for a given use case diagram. Our method incorporates two sets of rules for writing UCDs. Our first rule set, the syntax rules, describes the syntactic guidelines of sentences. Our second rule set, the step rules, shows the guidelines for writing each step specifically in UCDs. Our method also includes the three-level hierarchical use case evaluation approach– from the overview level, the use case element level, and the sentence level. The recommended techniques of our paper is a synthesis of a thorough comparison of various UCD contents and formats discussed in literatures and our own experiences developed through a graduate Systems Analysis and Design class for many years. We believe the methodology we proposed could serve as guidelines for UCD developers and help them to generate higher quality UCDs.

REFERENCES

1. Overgaard, G. and Palmkvist, K.: Use Cases: Patterns and Blueprints. Addison Wesley, (2005).
2. Adolph, S. and Bramble, P.: Patterns for Effective Use Cases. Addison Wesley, (2003).
3. Ben Achour, C., Rolland, C., Maiden, N., and Souveyet, C: Guiding use case authoring: Resultsof an empirical study. Proc. Fourth IEEE Int. Symposium on Requirements Engineering University of Limerick, Ireland, (1998). Retrieved 3/30/2006 from http://sunsite.informatik.rwth-aachen.de/CREWS/reports.htm. Report Number: 93-31.
4. Phalp, K.T., Vincent, J.V., and Cox, K.: Assessing the Quality of Use Case Descriptions, Accepted for the Software Quality Journal, February 2006, (2006) Retrieved 3/30/2006 from http://dec.bournemouth.ac.uk/ESERG/kphalp/
5. Cox, K. and Phalp, K.T.: Replicating the CREWS Use Case Authoring Guidelines Experiment, Empirical Software Engineering Journal, Vol. 5(3). (2000) 245-268.
6. Cockburn, A.: Writing Effective Use Cases. Addison Wesley, (2001).
7. Kulak, D. and Guinery, E.: Use Cases: Requirements in Context. 2nd ed. Addison.Wesley, (2003).
8. Cox, K., Phalp, K., and Shepperd, M.: Comparing Use Case writing guidelines. REFSQ'2001 - 7th International Workshop on Requirements Engineering: Foundation for Software Quality, Interlaken, Switzerland, 4th-5th June, (2001)
9. Rombach, H. D., Carbon, R., and Trapp, M. Software Engineering Processes and Measurement Research Group (AGSE), University of Kaiserslautern. (2003). Retrieved 3/20/2006 from: https://wwwagse.informatik.uni-kl.de/teaching/se1lab/ss2003/SysAnfBearbeiten/ChecklistenMitPerspektiven_Analyse.pdf
10. Copeland, L., Use cases and Testing, Testing UML Models, Part 1, Retrieved 3/30/2006 from: http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=3428
11. Bittner, K. and Spence, I.: Use Case Modeling. Addison-Wesley,(2003).
12. Texel, PP. and Williams, C.B.: Use Cases combined with BOOCH OMT UML. Prentice Hall PTR, (1997)
13. Eriksson, H., Penker, M., Lyons, B.,and Fado, D.: UML 2 Toolkit. Wiley Publishing, int, (2004)
14. Leffingwell, D. and Widrig, D.: Managing Software Requirements. Second Edition. A Use Case Approach. Addison Wesley, (2003).
15. Larman, C. (2005). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall PTR.
16. Song, I.-Y.: Object-Oriented Analysis and Design Using UML: A Practical Guide. Pearson Publishing, Boston, MA (2004).