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 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 to assess UCDs. The three-level hierarchical use case evaluation method helps to design the method that can cover 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.

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 Philp 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 for UCDs and their corresponding contents. There are many different techniques for determining how well a UCD is written. The most popular techniques include the following:

- A UCD should be a meaningful response to that action.
- All verbs are in present tense format.
- Give explanations if necessary.
- Avoid pronouns if there is more than one actor.
- No adverbs or adjectives.
- Avoid negatives.
- There should be logical coherence throughout the description.
- When an action occurs there should be a meaningful response to that action.

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

<table>
<thead>
<tr>
<th>CREWS rules</th>
<th>CP rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
<td>Structure 1: Subject verb object.</td>
</tr>
<tr>
<td>Style 2: Avoid pronouns if there is more than one actor.</td>
<td>Structure 2: Subject verb object prepositional phrase.</td>
</tr>
<tr>
<td>Style 3: No adverbs or adjectives.</td>
<td>Structure 3: Subject passive.</td>
</tr>
<tr>
<td>Style 4: Avoid negatives.</td>
<td>Structure 4: Underline other use case names.</td>
</tr>
<tr>
<td>Style 5: Give explanations if necessary.</td>
<td></td>
</tr>
<tr>
<td>Style 6: All verbs are in present tense format.</td>
<td></td>
</tr>
<tr>
<td>Style 7: There should be logical coherence throughout the description.</td>
<td></td>
</tr>
<tr>
<td>Style 8: When an action occurs there should be a meaningful response to that action.</td>
<td></td>
</tr>
</tbody>
</table>
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 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 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 postconditions 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. 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. 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


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: Use 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].

Table 3. A summary of the rule sets

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

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 be done 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
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

<table>
<thead>
<tr>
<th>Table 4. The three hierarchical heuristics of UCDs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Overall</strong></td>
</tr>
<tr>
<td><strong>Elements</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td><strong>Sentence</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
0 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/proceeding-paper/guidelines-developing-quality-use-case/33137

Related Content

Further Development of an Application Framework for Computational Chemistry (AFCC) Applied to New Drug Discovery

Topological Properties of Multigranular Rough sets on Fuzzy Approximation Spaces

Modeling Rumors in Twitter: An Overview

A Novel Call Admission Control Algorithm for Next Generation Wireless Mobile Communication

Forecasting Exchange Rates: A Chaos-Based Regression Approach