

A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation

Tao Yue¹, Lionel C. Briand², and Yvan Labiche¹

¹ Carleton University, Software Quality Engineering Lab, 1125 Colonel By Drive
Ottawa, ON K1S 5B6, Canada
{tyue, labiche}@sce.carleton.ca

² Simula Research Laboratory & University of Oslo,
P.O. Box 134, Lysaker, Norway
briand@simula.no

Abstract. Use case modeling is commonly applied to document requirements. Use case specifications (UCSs) are usually structured, textual documents complying with a certain template. However, because they remain essentially textual, ambiguities are inevitable. In this paper, we propose a new use case modeling approach, which is composed of a set of well-defined restriction rules and a template. The goal is to reduce ambiguity and facilitate automated analysis, though the later point is not addressed in this paper. We also report on a controlled experiment which evaluates the restriction rules and template in terms of their ease of application during use case modeling, and the quality of the analysis models derived from them by trained individuals. Results show that the restriction rules are overall easy to apply and that our use case modeling approach results in significant improvements regarding the correctness of derived class diagrams and the understandability of UCSs.

Keywords: Use Case; Use Case Modeling; Use Case Template; Restriction Rules; Analysis Model; Controlled Experiment.

1 Introduction

Use case modeling, including use case diagrams and use case textual specifications, is commonly applied to structure and document requirements [6, 8, 10]. Use Case Specifications (UCS) are usually textual documents complying with a use case template that, though helping read and review use cases, inevitably contains ambiguities. In this paper, we propose a set of restriction rules and a use case template, which are based in part on the results of a thorough literature review [24]. The goal is to restrict the way users can document UCSs in order to reduce ambiguity and facilitate automated analysis to derive initial analysis models, which in the Unified Modeling Language (UML) [14] are typically at least composed of class and interaction diagrams, and possibly other types of diagrams and constraints.

The restriction rules and the use case template we specify should be applied during the requirements elicitation phase of use case-driven software development (e.g., [4]) in order to produce, to the extent possible, precise and unambiguous use case models. A use case diagram in UML [14], is used to represent relationships

among actors and use cases, the latter being described by UCSs. The restriction rules we defined are applied to restrict the way users can write UCSs; the use case template is a means to structure UCSs. With a use case model documented by applying our use case modeling approach, initial analysis models of higher quality can hopefully be derived with greater ease. This step is usually manually performed by system analysts but the derivation of an initial analysis model could potentially be automated. This paper, however, does not address automation but focuses on describing our use case modeling method and assessing empirically its limitations and benefits.

More precisely, by means of experimentation, we aim at assessing whether our restriction rules are easy to apply while developing use case models and whether the overall approach helps the designer generate higher quality analysis models. Our experiment, involving fully trained, senior undergraduate students, shows that the proposed use case modeling approach results into higher quality class diagrams and that restriction rules are perceived overall to be easily applicable.

The rest of the paper is organized as follows. The related work is reported in Section 2. In Section 3, we discuss our use case modeling approach: the use case template and the restriction rules. The experimental evaluation of these rules and the template is presented in Section 4 (experiment planning), Section 5 (experiment results and analysis), and Section 6 (threats to validity). We conclude in Section 7.

2 Related work

Various use case templates (e.g., [5, 9-12]) have been suggested in the literature to satisfy different application contexts and purposes. These templates share common fields such as: use case name, brief overall description, precondition, postcondition, basic flow, and alternative flows. In addition to capturing requirements, use cases can also facilitate the automated derivation of initial analysis models – one of our goals. The systematic review [24] we conducted to examine works that transform textual requirements into analysis models reveals that six approaches require use cases. Their proposed templates (e.g., [7, 13, 19]) are similar to conventional ones but with some variations to facilitate the process of automatically deriving analysis models.

The use case template we propose in this paper (Section 3.1) integrates elements from many related works. It contains fields similar to those encountered in conventional templates but also seeks to better specify the structure of the flow of events. The ultimate motivation is to reduce ambiguity when models are derived manually but also support the automated generation of analysis models. Given our goals, we made the following decisions: (1) We included fields commonly encountered in most templates; (2) Some of the fields (e.g., scope) proposed in the literature to capture requirements were excluded since they do not help deriving analysis models; (3) We excluded the fields (e.g., the three-column steps modeling style proposed in [7]) that, on the one hand may increase the precision of UCSs but, on the other hand require that the designer provide much more information than in the end we do not need for our purpose. In other words, we believe that the additional precision does not warrant the additional cost, and that these fields do not bring clear advantages with respect to our objectives; (4) Six interaction types (five from [5], one we newly propose) are suggested to describe action steps in flows of events; (5) Differing from most of existing use case templates that suggest having one

postcondition for one use case, our template enforces that each flow of events (both basic flow and alternative flows) of a UCS contains its own postcondition.

In [24], we summarize and classify the restriction rules applied in [18-20, 22], which propose transformations from requirements to analysis models. Some guidelines on writing UCSs are also provided in various sources (e.g., [1, 3, 5]), based on practitioners' experience and to reduce ambiguities in UCSs or to facilitate the process of (automatically) deriving analysis models from them. In this paper, we propose a total of 26 restriction rules on the use of natural language to document UCSs that complies with our use case template. None of the related works we looked at relies on a set of rules as complete as the one we suggest. We reused some of the existing rules, excluded others, recommended new ones, and classified all the rules. Additionally, we explicitly describe why each of our restriction rules is needed either to reduce ambiguities or facilitate the process of (automatically) deriving analysis models, a crucial piece of information that is often omitted in the literature. We also indicate how and where to apply (Section 3.2) each of our restriction rules, another piece of information often left out by most papers on the topic. Several rules we newly propose in this work are based on our experience with several natural language parsers (e.g., [21]) and are proposed because sentences with certain structures cannot be correctly parsed. These rules can also help reduce ambiguity of UCSs and therefore help to manually derive analysis models from them. Furthermore, as opposed to many related works, our restriction rules are integrated with our use case template together as a comprehensive solution for use case modeling: several of our restriction rules refer to some of the features of our use case template (Section 3.1).

Some empirical studies (e.g., [1, 2, 16]) evaluated the impact of applying restriction rules on the quality of UCSs in terms of, for example, their completeness, structuredness and understandability. Results showed that using restriction rules led to more complete and better structured UCSs. These experiments evaluated restriction rules as a whole only, whereas we evaluate our restriction rules both individually and as a whole. By doing so, we can tell which rule(s) are difficult to apply and therefore require extra focus during training. It is also worth noticing that previous works assess the quality of UCSs against some quality criteria (e.g., understandability, structuredness, completeness), rather than test the ability of individuals to extract relevant information from UCSs to derive analysis models. In this paper, we report on a controlled experiment which evaluates the impact of our restriction rules and template both on the understandability of UCSs and the quality of analysis models generated from them in terms of correctness, completeness, and redundancy.

3 Use case modeling approach

In this section, we describe our use case modeling approach including a use case template (Section 3.1) and a set of restriction rules (Section 3.2).

3.1 Use case template

Our use case template has eleven first-level fields (1st column in Table 1). The last four fields are decomposed into second-level fields (2nd column in the last four rows). The last column of each row explains the corresponding field(s). There is no need to

further discuss the first seven fields since they are straightforward and commonly encountered in many templates. Below we focus the discussion on the *Basic Flow* and *Alternative Flows* fields.

A basic flow describes a main successful path. It often does not include any condition or branching [12]. It is recommended to describe separately the conditions and branching in alternative flows. A basic flow is composed of a sequence of steps and a postcondition. Each UCS can only have one basic flow. *Alternative flows* describe all the other scenarios or branches, both success and failure. An alternative flow always depends on a condition occurring in a specific step in a flow of reference, referred to as *reference flow*, and that reference flow is either the basic flow or an alternative flow itself. The branching condition is specified in the reference flow by following restriction rules (R20 and R22—Section 3.2). We refer to steps specifying such conditions as *condition steps* and the other steps as *action steps*. Similarly to the basic flow, an alternative flow is composed of a sequence of numbered steps. The action steps can be one of the following five interactions (which are reused from [5] except for the fifth): 1) Primary actor \rightarrow system: the primary actor sends a request and data to the system; 2) System \rightarrow system: the system validates a request and data; 3) System \rightarrow system: the system alters its internal state (e.g., recording or modifying something); 4) System \rightarrow primary actor: the system replies to the primary actor with a result; 5) System \rightarrow secondary actor: the system sends requests to a secondary actor. All steps are numbered sequentially. This implies that each step is completed before the next one is started. If there is a need to express conditions, iterations, or concurrency, then specific keywords, specified as restriction rules in Section 3.2, should be applied.

We classify alternative flows into three types: specific, global, and bounded alternative flows. This classification is adapted from [3]. A *specific alternative flow* is an alternative flow that refers to a specific step in the reference flow. A *bounded alternative flow* is an alternative flow that refers to more than one step in the reference flow—consecutive steps or not. A *global alternative flow* (called *general alternative flow* in [3]) is an alternative flow that refers to any step in the reference flow. Distinguishing different types of alternative flows makes interactions between the reference flow and its alternative flows much clearer. For specific and bounded alternative flows, a RFS (Reference Flow Step) section, specified as rule R9, is used to specify one or more (reference flow) step numbers. Whether and where the flow merges back to the reference flow or terminates the use case must be specified as the last step of the alternative flow. Similarly to the branching condition, merging is specified by following restriction rules (R24 and R25—Section 3.2). By doing so, we can avoid potential ambiguity in UCSs caused by unclear specification of interactions between the basic flow and its corresponding alternative flows. Each alternative flow must have a postcondition (enforced by restriction rule R26—Section 3.2).

It is usual to provide a postcondition describing a constraint that must be true when a use case terminates. If the use case contains alternative flows, then the postcondition of the use case should describe not only what must be true when the basic flow terminates but also what must be true when each alternative flow terminates. The branching condition to each alternative flow is then necessarily part of the postcondition (to distinguish the different possible results). In such a case, the postcondition becomes complex and the branching condition for each alternative flow

is redundantly described (both in the steps of flows and the postcondition), which therefore increases the risk of ambiguity in UCSs. Our template enforces that each flow of events (both basic flow and alternative flows) of a UCS contains its own postcondition and therefore avoids such ambiguity.

Table 1. Use case template

Use Case Name	The name of the use case. It usually starts with a verb.	
Brief Description	Summarizes the use case in a short paragraph.	
Precondition	What should be true before the use case is executed.	
Primary Actor	The actor which initiates the use case.	
Secondary Actors	Other actors the system relies on to accomplish the services of the use case.	
Dependency	Include and extend relationships to other use cases.	
Generalization	Generalization relationships to other use cases.	
Basic Flow	Specifies the main successful path, also called “happy path”.	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the basic flow executes.
Specific Alternative Flows	Applies to one specific step of the reference flow.	
	RFS	A reference flow step number where flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.
Global Alternative Flows	Applies to all the steps of the reference flow.	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.
Bounded Alternative Flows	Applies to more than one step of the reference flow, but not all of them.	
	RFS	A list of reference flow steps where flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow executes.

3.2 Restriction rules

The restriction rules are classified into two groups: restrictions on the use of natural language, and restrictions enforcing the use of specific keywords for specifying control structures. The first group of restrictions is further divided into two categories according to their location of application (see below). Each restriction rule is assigned a unique number.

Restriction rules R1-R16 in Table 2 constrain the use of natural language: the table explains why they are needed to reduce ambiguity. Rules R1-R7 apply only to action steps; they do not apply to condition steps, preconditions or postconditions. Rules R8-R16 apply to all sentences in a UCS: action steps, condition steps, preconditions, postconditions, and sentences in the brief description. Rules R8-R11 and R16 aim to reduce ambiguity of UCSs; the remaining rules (R12-R15) can help reduce ambiguity and also facilitate automated generation of analysis models. Recall that, as we discussed in Section 1, facilitating automated derivation of initial analysis models from UCSs is also one of our goals, though this is not discussed in this paper. These two sets of restrictions are thought to be good practice for writing clear and concise UCSs (e.g., [3, 5, 17]) except for R13 and R15. We add these two rules

because we observed that negative adverbs, negative adjectives, and participle phrases are very difficult to parse by natural language parsers.

Table 2.Restrictions (R1-R16)

#	Description	Explanation
R1	The subject of a sentence in basic and alternative flows should be the system or an actor.	Enforce describing flows of events correctly. These rules conform to our use case template (the five interactions).
R2	Describe the flow of events sequentially.	
R3	Actor-to-actor interactions are not allowed.	
R4	Describe one action per sentence. (Avoid compound predicates.)	Otherwise it is hard to decide the sequence of multiple actions in a sentence.
R5	Use present tense only.	Enforce describing what the system does, rather than what it will do or what it has done.
R6	Use active voice rather than passive voice.	Enforce explicitly showing the subject and/or object(s) of a sentence.
R7	Clearly describe the interaction between the system and actors without omitting its sender and receiver.	
R8	Use declarative sentences only.	Commonly required for writing UCSs.
R9	Use words in a consistent way.	Keep one term to describe one thing.
R10	Don't use modal verbs (e.g., <i>might</i>)	Modal verbs and adverbs usually indicate uncertainty.
R11	Avoid adverbs (e.g., <i>very</i>).	
R12	Use simple sentences only.	
R13	Don't use negative adverb and adjective (e.g., <i>hardly, never</i>), but it is allowed to use <i>not</i> or <i>no</i> .	Reduce ambiguity and facilitate automated NL parsing.
R14	Don't use pronouns (e.g. <i>he, this</i>)	
R15	Don't use participle phrases as adverbial modifier. For example, the italic-font part of the sentence "ATM is idle, <i>displaying a Welcome message</i> ", is a participle phrase.	
R16	Use "the system" to refer to the system under design consistently.	Keep one term to describe the system; therefore reduce ambiguity.

The remaining ten restriction rules (R17-R26) constrain the use of control structures, except R26 that specifies that each basic flow and alternative flow should have its own postcondition. R17 and R18 specify keywords to describe use case dependencies include and extend. Sentences containing the keywords INCLUDE USE CASE and EXTENDED BY USE CASE are referred to as dependency sentences. R19 specifies keyword RFS, which is used in a specific (or bounded) alternative flow to refer to a step number (or a set of step numbers) of a reference flow that this alternative flow branches from. Rules R20-R23 specify the keywords used to specify conditional logic sentences (IF-THEN-ELSE-ELSEIF-ENDIF), concurrency sentences (MEANWHILE), condition checking sentences (VALIDATES THAT), and iteration sentences (DO-UNTIL), respectively. Keyword VALIDATES THAT (R22) specifies that a condition is evaluated by the system and must be true to proceed to the next step. This rule also requires that an alternative flow describing what happens when the validation fails (the condition does not hold) be described. Rules R24 and

R25 specify that an alternative flow ends with a step using either keyword ABORT or keyword RESUME STEP, thereby clearly specifying whether the flow returns back to the reference flow and where (using keyword RESUME STEP followed by a returning step number) or terminates (using keyword ABORT).

R17-R21 and R23 have been proposed in the literature and we reused them with some variation. R22, R24 and R25 are newly proposed in this work for the purpose of making the whole set of restrictions as complete as possible so that flows of events and interactions between the basic flow and the alternatives can be clearly and concisely specified. Applying this set of rules helps reducing ambiguity in UCSs, and also facilitates automated NL processing and the generation of analysis models, especially sequence diagrams.

The detailed description of all the 26 restriction rules is provided in [25].

4 Experiment Planning

In this section, we follow the experiment reporting template proposed in [23]. All aspects of the experiment we conducted to assess our use case template and restriction rules are described and justified.

4.1 Experiment Definition

We are interested in the applicability of the restriction rules, combined with the use case template we propose. We refer to a use case model with UCSs that follow our restriction rules and use case template as a *restricted* use case model. We are also interested in the impact of a restricted use case model on the quality of analysis models that are manually derived from it, for instance by following standard guidelines for building analysis models (e.g., [4]). Indeed, if the restriction rules actually reduce ambiguity, then such models should exhibit higher quality. The experiment objectives are: characterizing each restriction rule with respect to their applicability (Goal 1), and evaluating the restriction rules and the use case template with respect to their impact on quality of derived analysis models (Goal 2). The evaluation of Goal 1 is a necessary pre-requisite to the investigation of Goal 2 in order to ensure that the restriction rules can be applied at a reasonable level of correctness. If the result of the experiment for Goal 1 shows that the restriction rules are applicable, then reliable use case models can be produced and we can go further to evaluate whether these restriction rules have an impact on the quality of manually generated analysis models (class and sequence diagrams in our experiment). In this paper, we focus on the experiment for Goal 2. Due to space limitation, the detailed discussion of the experiment for Goal 1 is omitted but is however provided in [25] for reference. Most noticeably, results for Goal 1 indicate that our 26 restriction rules are easy to apply [25].

4.2 Context Selection and subjects

The context of the experiment is a 4th Software Engineering course at Carleton University, Ottawa, Canada. The subjects selected were the 34 students registered in this course. The students were all trained in UML-based, object-oriented software development over the three years prior to the experiment and had therefore received

substantial training. Additionally, a lecture was given to them regarding the restriction rules and the use case template before the experiment. One assignment was also designed for the students to practice the restriction rules and the template. The results of the assignment were used to group the students into two blocks and therefore ensure better homogeneity across the two groups involved in the experiment. The experiment plan had been reviewed and received clearance through the Carleton University's Research Ethics Committee.

4.3 Hypotheses Formulation

The experiment for Goal 2 has one independent variable *Method*, with two treatments: *UCM_R* and *UCM_UR*, respectively denoting the use or not of the restriction rules, and two dependent variables *CD* and *QC*, respectively denoting the quality of analysis class diagrams and the correctness of responses to a comprehension questionnaire. We therefore can formulate the following null hypotheses (H_0) to be tested for each dependent variable of the experiment for Goal 2: there is no significant improvement in terms of CD and QC when using restricted use case models. The alternative hypotheses (H_a) is then one-tailed and stated as: restricted use case models result in high quality analysis models or high correctness of responses to the comprehension questionnaire when compared to unrestricted use case models.

4.4 Experiment Design

As stated previously, an assignment was designed to train the students to apply our restriction rules and template. Individual feedback was given to each student and a solution to the assignment was also provided before the experiment was conducted. Based on the grades of the assignment preceding the experiment, we defined the following three blocks: grades B to A+, grades B- to F, and absent (ABS). The students were then divided into two groups: A and B. Each of the two groups was then randomly assigned students from the three blocks in nearly identical proportions.

The students were asked to perform two tasks over two laboratories (3 hours each). In Lab 1 (Task 1), the students in group A were asked to produce UCSs of the Video Store (VS) system by applying the restriction rules and the use case template, whereas the students in group B did the same task on the Car Part Dealer (CPD) system. This first task was designed to address Goal 1: recall that this is omitted from this paper due to space limitation (the interested reader is referred to [25]). In Lab 2 (Task 2), which is dedicated to Goal 2, we further divided the students of group A into groups A1 and A2, so that the students in A1 derive class and sequence diagrams from a restricted use case model for the CPD system, while the students in A2 do the same from the unrestricted use case model following a standard template [4] of the same system. The same strategy was followed for group B but using the VS system instead. These two sub-groupings follow the same blocking strategy as the one used to group the students for Task 1 into groups A and B. Note that we use different systems for the two labs for each group of students in order to avoid learning effects that would otherwise constitute a threat to validity. For example, group A uses the VS system in Lab 1 but the CPD system in Lab 2. In total, 26 data points were obtained for Task 1 and Task 2 (14 data points for treatment *UCM_R* and 12 for treatment *UCM_UR*), respectively; however only 23 data points (14 for treatment *UCM_R* and

9 for treatment UCM_UR) were used for analyzing Task 2. Three data points were excluded from the analysis in order to avoid constituting a threat to validity: one student missed the lab for Task 2 and had to perform the task at home, another spent 3 hours 40 mins on Task 2, and the other produced a very incomplete result (no class diagram were derived).

4.5 Instrumentation

The instruments of an experiment are classified into three types: experiment objects, guidelines, and measurement instruments [23]. In this section we discuss our experiment instruments for Task 2 by conforming to this classification.

Experiment objects. The CPD and VS systems come in two versions for this part of the experiment: they contain the same use case diagram but have different UCSs (with or without restrictions). Both sets of UCSs—one with our restrictions and template, one with a standard template—were created by the first author of this paper. Both use case model versions were carefully reviewed by the authors to ensure that they contained equivalent information. Notice that the students were equally trained to understand our use case template and the standard template.

Experiment guidelines. A lab description was provided to the students at the beginning of each lab, describing the list of documents provided, the task of the lab, and the submission guidelines. The students belonging to different groups were monitored to ensure they would not access each other's documents during the entire lab duration. With a use case model as input documents, the students were asked to design a class diagram. We made it clear in the lab description that the students should, based on the use case description, assign meaningful names for each class, attribute, and operation, and apply the traditional *Entity/Boundary/Control* stereotype classification for each class. The students were also asked to complete a comprehension questionnaire during the lab, which was designed to evaluate how well they were able to understand the flow of events of each UCS. The students were also asked to derive sequence diagrams for two selected use cases; however most of the students were not able to derive these diagrams due to time constraints, which were therefore not analyzed.

Measurement instruments. A comprehension questionnaire was designed for each system to quickly evaluate, in a repeatable and objective way, the extent to which students understood the main body (flows of events) of each UCS. The standard guidelines proposed in [15] were followed to create the questionnaires. To avoid introducing any bias, we ensured comprehension questions were answerable by the students using both the restricted or unrestricted use case models. The complete questionnaires for the two systems are discussed in [25].

4.6 Evaluation measurement and data collection

There are two dependent variables for Goal 2, for which data must be collected in Task 2: the quality of class diagrams (abbreviated as CD) and the correctness of responses to the comprehension questionnaires (abbreviated as QC).

Variable CD. The quality of an analysis class diagram is evaluated from three aspects: *Correctness*, *Completeness*, and *Redundancy*. We used reference class diagrams, designed by the authors, as the basis to evaluate class diagrams designed by the

students. Data are collected from the reference class diagrams (i.e., number of classes, associations and generalizations), and data are also collected from the class diagrams of each student (e.g., number of missing classes, missing attributes of a class, incorrect associations, and redundant classes). All these data are then used to compute the measures of *Completeness*, *Correctness* and *Redundancy* of a student class diagram. The completeness of a class diagram is inversely related to the numbers of missing classes, associations and generalizations, which are considered to be three important element types in a class diagram; the correctness of a class diagram is determined by the correctness of matching classes (computed as the average, over the complete class diagram, of the class measures of *Completeness* and *Correctness*) and associations; the redundancy of a class diagram is computed as the ratio of redundant classes over all the classes of a student's class diagram. The completeness of a class is related to whether its stereotype is missed and whether there are missing attributes and operations; the correctness of a class is determined by whether the class is correctly named, stereotyped and specified as abstract, and whether a single logical concept is represented and a cohesive set of responsibilities is assigned to the class. The detailed description of the measures and calculation formulas is provided in [25] due to space limitations.

For each reference class of the reference class diagram of a case study system, we look for a class with the same name as the reference class in a student class diagram. If such a class is found, then this matching class is evaluated according to the quality measures for a class; otherwise, we keep looking in the student class diagram for a design equivalent¹ to the reference class. If no such equivalent design exists in the student class diagram, then we identify the reference class as missing and therefore the student diagram as incomplete. When all the reference classes in the reference class diagram have been looked at, there are three outputs: 1) a set of matching classes are identified and evaluated by using the quality measures for a class; 2) a set of equivalent designs are identified but not measured because either a subjective measurement or a large number of specific measures would be required to measure them. Besides, not many such equivalent designs have been found and not measuring them does not really impact the measurement of CD; and 3) a set of reference classes, missing in the student class diagram (i.e., not matching classes or equivalent design), are listed. A procedure similar to this identification of matching class, missing class, and equivalent class designs is also applied to identify matching/missing attributes, operations, associations, and generalizations.

Variable QC. Data about the correctness of responses to the questions of the comprehension questionnaires of Task 2 are used to evaluate the understandability of UCSs, which is normalized between 0 and 1:

$$\text{For the CPD system: } \quad QC_{CPD} = \text{number of correct responses} / 15$$

$$\text{For the VS system: } \quad QC_{VS} = \text{number of correct responses} / 25$$

where the denominators are the total numbers of questions in each system questionnaire.

¹ An equivalent design may contain one or more model elements, which could be attributes, multiple classes connected by associations, etc.

5 Experiment Results and analysis

In this section, we present the results from the controlled experiment described in the previous section. Though the two systems used for the experiment might lead to different results, the number of observations does not allow us to perform a separate analysis for each of them. We, however, counter-balance their possible effect by ensuring a similar proportion of observations coming from each system, for each of the tasks. Recall that Goal 1 is to evaluate whether our use case modeling approach (the use case template and the restriction rules) is easy to apply while developing use case models. Each restriction rule is evaluated in terms of its understandability, applicability, restrictiveness, and error rate. Though we do not report these results in detail here, they indicate that our 26 restriction rules are easy to apply and with appropriate tool support and focused training on the rules receiving higher error rates, error rates can be expected to decrease, as detailed in [25]. Based on these results, we are therefore confident that trained engineers are capable to properly apply our restriction rules and template and obtain UCSs from which to derive analysis models. In the rest of the section, we report the experiment results for Goal 2 (Task 2).

As we have discussed in Section 4.3, Goal 2 involves one independent variable (*Method*) with two treatments, *UCM_R* and *UCM_UR*, respectively denoting the use or not of restriction rules, and two dependent variables *CD* and *QC*, respectively denoting the quality of analysis class diagrams and the correctness of responses from a comprehension questionnaire. In this section, we report on one-tailed *t*-test results using the factor *Method*.

Table 3. Descriptive statistics of all measures

Methods	Completeness		Correctness		Redundancy		QC	
	Mean	Size	Mean	Size	Mean	Size	Mean	Size
UCM_R	0.260	14	0.882	14	0.093	14	0.913	12
UCM_UR	0.178	9	0.807	9	0.141	9	0.527	8
All Methods	0.219	23	0.845	23	0.117	23	0.72	20

The descriptive statistics of all measures are presented in Table 3. As shown in this table, all means for *Completeness* are below 0.3. This means that less than 30% of required class diagram elements (e.g., classes, associations, and generalizations) were derived from UCSs by the students. This is likely due to time constraints of the experiment. All means for *Redundancy* are below 0.15, which indicates that student-derived class diagrams have very low redundancy². This can also be explained by time constraints during the experiment: the students were not able to completely design class diagrams (low *Completeness*) and there was therefore less opportunity to define redundant classes. *Correctness* evaluates each matching class and association³ in the students' class diagrams; therefore time constraints have no impact on the results of this measure. This statement is also supported by the data shown in Table 3: All *Correctness* means are above 0.8, which means a 80% *Correctness* in the matching classes and associations of the students' class diagrams. The QC

² Only redundant classes are used to measure *Redundancy* of class diagrams.

³ Missing classes and associations are taken care of by *Completeness* and redundant classes are measured by *Redundancy*.

(Questionnaire) mean of treatment UCM_R is over 90%, which means that time constraints had little impact on QC results: the students had enough time to correctly answer over 90% of the questions. The students with treatment UCM_UR only correctly answered 52.7% of the questions during the same period of time. The significant difference between the two treatments is what we expected (i.e., restrictions helped), which is analyzed next.

Table 4 presents a summary of the statistical *t*-test results for dependent variables CD and QC. Regarding CD, the students with treatment UCM_R performed slightly better in terms of *Completeness* and *Redundancy* than otherwise, but the difference is not statistically significant, due perhaps to time constraints of the experiment (both of the two measures received low mean values as shown in Table 3) and the small size of our sample. However, there is a statistically significant difference regarding *Correctness*: the students with treatment UCM_R produced significantly higher quality class diagrams than the students with treatment UCM_UR. Regarding QC, the *t*-test result also shows a significant difference between the two treatments in the expected direction, thus indicating an increased understanding due to restriction rules and the template. The magnitude of the difference is also very large: 38.7% (Table 4). Nonparametric tests were also performed. The results are not very different from the *t*-test results and are therefore not presented in this paper.

Table 4. *t*-test – CD and QC

Measures	Mean difference (UCM_R – UCM_UR)	DF	t-value	p-value
Completeness	0.082	17	1.552	0.0695
Correctness	0.074	17	2.348	0.0155
Redundancy	-0.048	12	-0.792	0.2218
QC	0.387	8	5.189	<0.0004

As stated previously, statistically significant differences are obtained in terms of both *Correctness* and QC (differences of 0.074 and 0.387, respectively—Table 4). The difference in size between the effect on *Correctness* and QC can also be explained. As discussed in Section 3.2, R1-R7 put restrictions on the use of natural language but can only be applied to action steps; R8-R16 also put restrictions on the use of natural language but can be applied to both action steps and condition steps; R17-R25 are rules on the use of control structures specified as keywords. By looking at those rules, it appears that R1-R7 and R17-R25 primarily put restrictions on documenting flows within steps (sentences) or flows of steps in UCSs, while R8-R16 are more related to the vocabulary being used in all the sentences of a UCS. Therefore we believe that R8-R16 impacted the quality of derived class diagrams (CD) to a larger extent than the other rules. On the other hand, we believe that rules R1-R7 and R17-R25 had a greater impact on the result of comprehension questionnaires (QC) than R8-R16, since questionnaires evaluated the extent to which students understood the flows of events of each UCS. Then because a much larger number of restriction rules have an impact on QC than CD, the mean differences between the two treatments in terms of CD *Correctness* and QC are likely to reflect that difference of impact: the mean difference in QC is much larger than the mean difference in CD *Correctness*. This intuition could perhaps be confirmed by studying the quality of

interaction diagrams generated by students: since they relate more to flow than vocabulary we would expect their quality to be higher. However, recall that due to time constraints, our students did not have time to produce interaction diagrams. This will be the purpose of future work.

6 Threats to Validity

Two main threats to external validity are relevant to our experiment, and are typical of what can be found when running controlled experiments in artificial settings and within time constraints: 1) Are the subjects representative of software professionals? 2) Is the experiment material representative of industrial practice?

Regarding issue 1), recall that in Task 1 the students designed use case models by applying our restriction rules and use case template. This task is usually performed by requirements engineers during the requirements elicitation phase of a typical software development lifecycle. Given the state of practice in most of the software industry, whether for students or professional requirements engineers, it is likely to require training. The students of our experiment are 4th year software and computer engineering students who had received training in use case modeling in previous courses. In addition, they were given a 90 minute lecture and an assignment specifically focusing on how to apply the restriction rules and template. In our context, the main difference between students and professional requirements engineers, is that the latter could have more experience on designing use case models, and thus we assume that they would probably apply more effectively our use case modeling approach than students given the same amount of training. Thus, professional requirements engineers would be able to further benefit from the restriction rules and template, and thus provide a more positive opinion on the rules' applicability. As for Task 2, the students derived analysis models from both the restricted and unrestricted use case models. This task is usually performed by system analysts in industry. Again, our 4th year software and computer engineering students had received extensive training on software modeling with the UML, through several courses, and this is more than what we have observed in most software development environments.

As for issue 2) above, the scale of the systems is not likely to have a significant impact on the results of the experiment for Task 1. Indeed, this task does not require an overall understanding of the systems as the use case diagrams of the two systems were provided to the students as part of the experiment material. The students were only asked to write some UCSs by applying the restriction rules and the template. Due to time constraints (two three-hour laboratories), it was anyway not feasible to consider larger scale systems (with more UCSs) for Task 2.

Construct validity is related to our measurement instruments: the two comprehension questionnaires used respectively for the two tasks. The questions of the comprehension questionnaire for Task 2 are designed to be answerable from the use case models with or without restrictions, therefore introducing no bias for any of the treatments. Three students presented problems related to internal validity. One of them missed the lab for Task 2 and had to perform the task at home; another spent 3 hours 40 mins on Task 2; the other produced a very incomplete result (no class diagram was derived). These three data points were excluded from the analysis.

7 Conclusion

Use case modeling is one of the most common practices for capturing functional requirements. However, use case specifications (UCSs) are essentially textual documents and therefore ambiguity is inevitably introduced. To facilitate the transition towards analysis models, whether manual or automated, the UCSs are expected to be the least ambiguous possible. In this paper, we propose a use case modeling approach, which is composed of 26 well-defined restriction rules and a use case template, to restrict the way users can document UCSs in order to reduce ambiguity and also facilitate the (automated) transition towards analysis models.

A controlled experiment was conducted, in the context of a 4th year Software Engineering course, to evaluate whether our use case modeling approach is easy to apply while developing use case models and whether it helps obtain higher quality analysis models. Each restriction rule was evaluated in terms of its understandability, applicability, restrictiveness, and error rate. Though not presented in detail here, the experiment results indicate that our 26 restriction rules are easy to apply and can therefore help obtain UCSs that are a reliable source from which to derive analysis models. This was a prerequisite to the investigation reported in this paper.

The second part of the controlled experiment, presented in detail here, was to evaluate whether our use case modeling approach helps derive higher quality analysis models, by comparing it to a common use case modeling approach that does not put restrictions on natural language. The quality of analysis class diagrams is evaluated from the viewpoints of their correctness, completeness, and redundancy. The results show that our use case modeling approach leads to significant improvements regarding the correctness of derived class diagrams, but not their completeness and redundancy. We believe this is likely due to the time constraints of the experiment; the students were not even close to complete the class diagrams and there was therefore less opportunity to define redundant classes. Furthermore, our approach resulted in a large improvement in term of the students' comprehension of the use case model as measured by a carefully designed questionnaire.

Based on our knowledge, this study represents the first controlled experiment that evaluates the applicability of restriction rules used to document UCSs both individually and as a whole and that also evaluates the impact of these rules and template on the quality of generated analysis class diagrams. The measures we have defined to characterize restriction rules and evaluate the quality of analysis class diagrams can be reused for similar experiments to be conducted in the future.

During the second part of the experiment, the students were also asked to derive sequence diagrams for two use cases. However, most of the students were not able to do so due to time constraints. Evaluating the impact of our use case modeling approach on the quality of analysis sequence diagrams would be a valuable future work. In addition, we also plan to replicate the experiment to see whether significant differences between two treatments can be identified in terms of completeness and redundancy of generated analysis class diagrams if more time is given to participants of the experiment.

8 References

- [1] Achour C. B., Rolland C., Maiden N. A. M. and Souveyet C., “Guiding use case authoring: Results of an empirical study,” *Proc. RE'99*, pp. 36-43, 1999.
- [2] Anda B., Sjoberg D. and Jorgensen M., “Quality and understandability of use case models,” *Proc. ECOOP 2001*, pp. 402-428, 2001.
- [3] Bittner K. and Spence I., *Use Case Modeling*, Object Technology, Addison-Wesley Boston, 2002.
- [4] Bruegge B. and Dutoit A. H., *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 2nd Edition, 2004.
- [5] Cockburn A., *Writing effective use cases*, Addison-Wesley Boston, 2001.
- [6] Dobing B. and Parsons J., “How UML is used,” *Communications of the ACM*, vol. 49 (5), pp. 109-113, 2006.
- [7] Insfrán E., Pastor O. and Wieringa R., “Requirements Engineering-Based Conceptual Modelling,” *Requirements Engineering*, vol. 7 (2), pp. 61-72, 2002.
- [8] Jacobson I., “Use cases - yesterday, today, and tomorrow,” *Software and Systems Modeling*, vol. 3 (3), pp. 210-220, 2004.
- [9] Jacobson I., Christerson M., Jonsson P. and Overgaard G., *Object-oriented software engineering: a use case driven approach*, Addison-Wesley, 1992.
- [10] Kruchten P., *The Rational Unified Process: An Introduction*, Addison-Wesley, 2003.
- [11] Kulak D. and Guiney E., *Use cases: requirements in context*, ACM Press, 2000.
- [12] Larman C., *Applying UML and Patterns*, Prentice-Hall, 3rd Edition, 2004.
- [13] Liu D., *Automating Transition from Use Cases to Class Model*, Thesis, University of Calgary, Department of Electrical and Computer Engineering, 2003
- [14] OMG, “UML 2.0 Superstructure Specification,” Object Management Group, <http://www.omg.org/spec/UML/2.0/>, 2005.
- [15] Oppenheim A. N., *Questionnaire design, interviewing, and attitude measurement*, Pinter Pub Ltd, 1992.
- [16] Phalp K. T., Vincent J. and Cox K., “Improving the quality of use case descriptions: empirical assessment of writing guidelines,” *Soft. Quality Journal*, 15 (4), pp. 383-399, 2007.
- [17] Schneider G. and Winters J. P., *Applying use cases: a practical guide*, Object Technology, Addison-Wesley, 1998.
- [18] Śmiałek M., Bojarski J., Nowakowski W., Ambroziewicz A. and Straszak T., “Complementary Use Case Scenario Representations Based on Domain Vocabularies,” *Proc. MoDELS*, 2007.
- [19] Somé S. S., “Supporting use case based requirements engineering,” *Information and Software Technology*, vol. 48 (1), pp. 43-58, 2006.
- [20] Subramaniam K., Liu D., Far B. H. and Eberlein A., “UCDA: Use Case Driven Development Assistant Tool for Class Model Generation,” *Proc. SEKE'04*, 2004.
- [21] The Stanford Parser version 1.6: The Stanford Natural Language Processing Group, <http://nlp.stanford.edu/software/lex-parser.shtml>.
- [22] Wahono R. S. and Far B. H., “A framework for object identification and refinement process in object-oriented analysis and design,” *Proc. Cognitive Informatics*, pp. 351-360, 2002.
- [23] Wohlin C. and Wesslen A., *Experimentation in Software Engineering: An Introduction*, Springer, 2000.
- [24] Yue T., Briand L. C. and Labiche Y., “A Systematic Review of Transformation Methodologies between User Requirements and Analysis Models,” Carleton University, Technical Report SCE-09-03, 2009.
- [25] Yue T., Briand L. C. and Labiche Y., “A Use Case Modeling Approach to Facilitate the Transition Towards Analysis Models: Concepts and Empirical Evaluation,” Carleton University, Technical Report SCE-09-05, 2009.