Facilitating the Transition from Use Case Models to Analysis Models: Approach and Experiments

TAO YUE, Simula Research Laboratory, Norway
LIONEL C. BRIAND, Simula Research Laboratory and University of Oslo
YVAN LABICHE, Carleton University, Canada

Use case modeling, including use case diagrams and use case specifications (UCSs), is commonly applied to structure and document requirements. UCSs are usually structured but unrestricted textual documents complying with a certain use case template. However, because Use Case Models (UCMods) remain essentially textual, ambiguity is inevitably introduced. In this article, we propose a use case modeling approach, called Restricted Use Case Modeling (RUCM), which is composed of a set of well-defined restriction rules and a modified use case template. The goal is two-fold: (1) restrict the way users can document UCSs in order to reduce ambiguity and (2) facilitate the manual derivation of initial analysis models which, when using the Unified Modeling Language (UML), are typically composed of class diagrams, sequence diagrams, and possibly other types of diagrams.

Though the proposed restriction rules and template are based on a clear rationale, two main questions need to be investigated. First, do users find them too restrictive or impractical in certain situations? In other words, can users express the same requirements with RUCM as with unrestricted use cases? Second, do the rules and template have a positive, significant impact on the quality of the constructed analysis models? To investigate these questions, we performed and report on two controlled experiments, which evaluate the restriction rules and use case template in terms of (1) whether they are easy to apply while developing UCMods and facilitate the understanding of UCSs, and (2) whether they help users manually derive higher quality analysis models than what can be generated when they are not used, in terms of correctness, completeness, and redundancy. This article reports on the first controlled experiments that evaluate the applicability of restriction rules on use case modeling and their impact on the quality of analysis models. The measures we have defined to characterize restriction rules and the quality of analysis class and sequence diagrams can be reused to perform similar experiments in the future, either with RUCM or other approaches.

Results show that the restriction rules are overall easy to apply and that RUCM results into significant improvements over traditional approaches (i.e., with standard templates, without restrictions) in terms of class correctness and class diagram completeness, message correctness and sequence diagram completeness, and understandability of UCSs.

Categories and Subject Descriptors: D.2.1 [**Software Engineering**]: Requirements/Specification—*Methodologies*

General Terms: Design, Documentation, Experimentation, Measurement

Additional Key Words and Phrases: Use case, use case modeling, use case template, restriction rules, analysis model, class diagram, sequence diagram, controlled experiment.

L. C. Briand is currently affiliated with the SnT Centre, University of Luxembourg.
L. C. Briand was supported by a PEARL grant from the Fonds National de la Recherche, Luxembourg (FNR).
This article is a revised and extended version of a paper presented at MODELS'09.
Authors' addresses: T. Yue, Simula Research Laboratory, Olso, Norway; email: tao@simula.no; L. C. Briand, SnT Centre, University of Luxembourg; email: lionelbriand@uni.lu; Y. Labiche, Software Quality Engineering Laboratory, Carleton University, Ottawa, Canada; email: labiche@sce.carleton.ca.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2013 ACM 1049-331X/2013/02-ART5 $15.00
DOI 10.1145/2430536.2430539 http://doi.acm.org/10.1145/2430536.2430539

ACM Reference Format:
Yue, T., Briand, L. C., and Labiche, Y. 2013. Facilitating the transition from use case models to analysis models: approach and experiments. ACM Trans. Softw. Eng. Methodol. 22, 1, Article 5 (February 2013), 38 pages.
DOI = 10.1145/2430536.2430539 http://doi.acm.org/10.1145/2430536.2430539

1. INTRODUCTION

Use case modeling, including use case diagrams and use case textual specifications, is commonly applied to structure and document requirements [Dobing and Parsons 2006; Jacobson 2004; Kruchten 2003]. Use Case Specifications (UCSs) are usually textual documents complying with a use case template that, though helping reading and reviewing use cases, inevitably contain ambiguities. In this article, we propose a set of restriction rules and a use case template, which are based in part on the results of a thorough systematic literature review [Yue et al. 2011b]. The goal is to restrict the way users can document UCSs in order to reduce ambiguity and facilitate the construction of initial analysis models, either manually or automatically. Such models, in the context of the Unified Modeling Language (UML) [OMG 2005], are typically at least composed of class and sequence diagrams, and possibly other types of diagrams and constraints. Our use case modeling approach is denoted as Restricted Use Case Modeling (RUCM).

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., Bruegge and Dutoit [2009]) in order to produce, to the extent possible, precise and unambiguous Use Case Models (UCMods). A use case diagram in UML [OMG 2005] is used to represent relationships among actors and use cases described by UCSs. The restriction rules are applied to restrict the way users can write UCSs; the use case template is a means to structure UCSs. With a UCMod documented by applying RUCM, initial analysis models of higher quality can hopefully be derived with greater ease. This step is usually performed manually by system analysts but recent work has shown it can also be *partially* automated [Yue et al. 2010b], though presenting and studying this kind of automation is out of the scope of this article.

Though the restriction rules we proposed are based on a clear rationale, users might find them too restrictive or impractical in certain situations. They must therefore be experimentally evaluated in order to assess whether (1) they are easy to apply while developing UCMods, (2) they help the designer construct higher quality analysis models than what can be generated when they are not used, and (3) they can help *automatically* deriving analysis models from UCMods. This article investigates the first two questions through controlled experiments with fully trained, 4th year undergraduate students. We already addressed the third question in Yue et al. [2011a, 2010a, 2010b] where a tool called aToucan is proposed to automatically derive an analysis model including class, sequence, activity, and state machine diagrams, from a RUCM model. Such transformations are however incomplete and the manual construction of models is therefore still relevant.

Several empirical studies (e.g., Achour et al. [1999], Anda et al. [2001], and Phalp et al. [2007b]) have been conducted to examine the applicability of restriction rules; however all these experiments evaluated restriction rules as a whole instead of evaluating their individual impact. Note that the set of rules evaluated by these approaches is not the same and the purpose neither. In this article, we report on the first controlled experiments to evaluate the individual and combined applicability of restriction rules during use case modeling. One benefit of doing so is that it becomes feasible to precisely tell which rule(s) are difficult to apply and should therefore receive more attention during training. Other approaches empirically evaluate the role of use cases in the construction of class diagrams [Anda and Sjoberg 2005], empirically assess a use case construction and defect detection approach [Belgamo et al. 2005], or subjectively compare different approaches to manually derive class diagrams from use cases [Liang 2003]. These approaches have different objectives than ours, which is to evaluate the impact of RUCM on the quality of analysis models generated from RUCM models. This is of high practical importance as such models are then used to support the design process. The measures we have defined to characterize restriction rules and the quality of analysis class and sequence diagrams can be reused to perform similar experiments in the future, either with RUCM or other approaches.

This article is an extension of Yue et al. [2009], where we first described RUCM and reported on an initial controlled experiment to evaluate its applicability and impact on the quality of manually derived analysis models. The extensions in the current article are as follows: 1) A replication of the experiment reported in Yue et al. [2009], 2) An assessment of the impact of RUCM on manually derived sequence diagrams in addition to class diagrams, 3) Due to space limitation in the conference paper [Yue et al. 2009], the design and analysis of the initial experiment were partially reported. In this article, we report them in much greater details, thus facilitating future replications and comparisons.

The rest of the article is organized as follows. Section 2 discusses RUCM: the use case template and the restriction rules. The experimental evaluation of RUCM is presented in Section 3 (experiment planning), and Section 4 (experiment results and analysis). In Section 5, we report on the threats to validity and discuss practical implications and limitations of our research. Related work is reported in Section 6, and we conclude in Section 7.

2. USE CASE MODELING APPROACH (RUCM)

In the context of UML-based development, a UCMod, modeling the functional requirements of a system, is composed of a UML use case diagram and a set of structured, textual UCSs. Each use case of a use case diagram is described in a textual UCS complying with a certain use case template. We specify a new use case template (Section 2.1) that merges several aspects of existing templates. We also propose a set of restrictions to the use of plain language for documenting UCSs (Section 2.2).

2.1. Use Case Template

Our use case template has eleven first-level fields (first column of Table I). The last four fields are decomposed into second-level fields (second column of 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 (Section 6.1). Below we focus the discussion on the *Basic Flow* field and the three types of *Alternative Flows*: specific, global, and bounded alternative flows.

A *basic flow* describes a main successful path. It often does not include any condition or branching [Larman 2004]. We recommend describing 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. The action steps can be one of the following five interactions, which are reused from Cockburn [2001] except for the fifth:

(1) Primary actor→system: the primary actor sends a request and data to the system.
(2) System→system: the system validates a request and data.
(3) System→system: the system alters its internal state (e.g., recording or modifying something).
(4) System→primary actor: the system replies to the primary actor with a result.
(5) System→secondary actor: the system sends requests to a secondary actor.

Table I. RUCM 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 basic 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 basic 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 basic 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.

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 2.2, should be applied.

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 2.2). 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. We classify alternative flows into three types: specific, global, and bounded alternative flows. This classification is adapted from Bittner and Spence [2002].

(1) A *specific alternative flow* is an alternative flow that refers to a specific step in the reference flow.
(2) A *bounded alternative flow* is an alternative flow that refers to more than one step in the reference flow–consecutive steps or not.
(3) A *global alternative flow* (called *general alternative flow* in Bittner and Spence [2002]) 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 R19 (Section 2.2), is used to specify one or more (reference flow) step numbers. Whether and where the flow of an alternative 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 and termination are specified by following restriction rules (R24 and R25—Section 2.2). By doing so, we can avoid potential ambiguity in UCSs, caused by unclear specification of interactions between the basic flow and its corresponding alternative

Table II. Restriction Rules (R1–R7)

#	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.	

flows. Each alternative flow must have a postcondition (enforced by restriction rule R26—Section 2.2).

It is usual to provide a postcondition describing a constraint that must be true when a use case terminates [Larman 2004]. 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 results). In such a case, the postcondition can become 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 (both basic flow and alternative flows) of a UCS contains its own postcondition and therefore avoids such complexity. An example of UCS documented with RUCM is presented in Table XXVI, Appendix A. (For comparison purposes, the same UCS, not documented with RUCM is also available in Appendix A: in Table XXVII.)

2.2. 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.

Seven restriction rules (R1–R7) constrain the use of natural language (Table II): the table explains why they are needed to reduce ambiguity. Notice that these rules apply only to action steps; they do not apply to condition steps or preconditions or postconditions. The other nine restriction rules (Table III) apply to all sentences in a UCS: action steps, condition steps, preconditions, postconditions, and sentences in the brief description. Rules R8–R10 and R16 are to reduce ambiguity of UCSs; the remaining rules specifically facilitate automated generation of analysis models, though they can also help reduce ambiguity. These two sets of restrictions have been discussed in the literature (e.g., Bittner and Spence [2002], Cockburn [2001], and Schneider and Winters [1998]), with the exception of R13 and R15, and have been thought to be good practice for writing clear and concise UCSs. We include R13 and R15 because we perceived that negative adverbs, negative adjectives, and participle phrases are very difficult to parse for natural language parsers. R9 requires using words consistently to

Table III. Restriction Rules (R8-R16)

#	Description	Explanation
R8	Use declarative sentence only. "Is the system idle?" is a non-declarative sentence.	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., *might*)	Modal verbs and adverbs usually indicate uncertainty; Instead, metrics should be used if possible.
R11	Avoid adverbs (e.g., *very*).	
R12	Use simple sentences only. A simple sentence must contain only one subject and one predicate.	Facilitate automated natural language parsing and reduce ambiguity.
R13	Don't use negative adverb and adjective (e.g., *hardly*, *never*), but it is allowed to use *not* or *no*.	
R14	Don't use pronouns (e.g. *he*, *this*)	
R15	Don't use participle phrases as adverbial modifier. For example, the italic-font part of the sentence "ATM is idle, *displaying a Welcome message*", 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.

Table IV. Restriction Rules (R17–R26)

#	Description	#	Description
R17	INCLUDE USE CASE	R22	VALIDATE THAT
R18	EXTENDED BY USE CASE	R23	DO-UNTIL
R19	RFS	R24	ABORT
R20	IF-THEN-ELSE-ELSEIF-ENDIF	R25	RESUME STEP
R21	MEANWHILE	R26	Each basic flow and alternative flow should have its own postconditions.

document UCSs. A common approach to do so is to use a domain model and glossary (e.g., Bruegge and Dutoit [2009], and Larman [2004]) as a basis to write UCSs.

The remaining ten restriction rules (Table IV) specify 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*. 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. The keyword IF-THEN-ELSE-ELSEIF-ENDIF can be used in three different ways (these are specified as a grammar): 1) IF-THEN-ENDIF (appears in one flow only), 2) IF-THEN-ELSE-ENDIF (everything is in one flow, or IF-THEN in one flow and ELSE in an alternative flow), and 3) IF-THEN-ELSEIF-THEN-ELSE-ENDIF (everything is in one flow, or IF-THEN in a flow and ELSEIF-THEN-ELSE-ENDIF in an alternative flow). Keyword VALIDATES THAT (R22) means that the condition is evaluated by the system and must be true to proceed to the next step. This rule also requires an alternative flow describing what happens when the validation fails (the condition does not hold). Rules R20 and R22 are complex rules, when compared with the others of the same rule set, because both of them require that UCS designers look at multiple steps in two different flows: the basic flow and an alternative flow.

R24 and R25 specify keywords ABORT and RESUME STEP to describe an exceptional exit action and when an alternative flow goes back to its corresponding basic flow, respectively. These two rules also specify that an alternative flow ends either with ABORT or RESUME STEP, which means that the last step of the alternative flow should clearly specify whether the flow returns back to the basic flow and where (using keywords RESUME STEP followed by a returning step number) or terminates (using keyword ABORT).

R17–R21 and R23 have been proposed in the literature [Somé 2006; Subramaniam et al. 2004] and we reused them with some variation. R22, R24 and R25 have been added 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 facilitates automated NL processing (e.g., correctly parse sentences with our specified keywords) and generating of analysis models, especially sequence diagrams which also helps reducing ambiguity of UCSs [Yue et al. 2011a; Yue et al. 2010a; Yue, et al. 2010b]. A more detailed description of all the 26 restriction rules is provided in a technical report [Yue 2010; Yue, et al. 2009].

3. EXPERIMENT PLANNING

In this section, we adapted the experiment reporting template proposed in Wohlin and Wesslin [2000]. All aspects of the experiments we conducted to assess our use case template and restriction rules are described and justified.

3.1. Experiment Definition

We are interested in the applicability of the restriction rules, combined with the use case template we proposed. We refer to a UCMod with UCSs that follow our restriction rules and template as a *restricted* UCMod. We are also interested in the impact of a restricted UCMod on the quality of analysis models that are manually derived from it, for instance by following standard guidelines for building analysis models (e.g., Bruegge and Dutoit [2009]). Indeed, if the restriction rules actually reduce ambiguity, then such models should exhibit higher quality.

The experiment objectives are formulated as Goal-Question-Metric (GQM) goals [Basili et al. 1994] as follows.

Goal 1. Analyze the restriction rules *for the purpose of* characterizing *with respect to* the applicability of each restriction rule *from the point of view of* the requirements engineer *in the context of* 4th year undergraduate students defining use case models.

Goal 2. Analyze the restriction rules and the RUCM use case template *for the purpose of* evaluating *with respect to* their impact on quality of derived analysis models *from the point of view of* the system analyst *in the context of* 4th year undergraduate students manually deriving analysis models from use case models.

The evaluation of Goal 1 is a necessary prerequisite to the investigation of Goal 2: we need 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 we can go further and investigate whether RUCM has an impact on the quality of manually generated analysis models (class and sequence diagrams in our experiment).

Regarding Goal 1, the applicability of each restriction rule is characterized in terms of *Error Rate*, *Understandability*, *Applicability*, and *Restrictiveness*. The experiment for Goal 2 has one independent variable, namely *Method*, with two treatments corresponding to the usage or not of our restriction rules and proposed template. The alternative was to use one of the well-known use case templates and no restriction rules in the textual description of UCSs. (The two treatments are illustrated on example use case in Appendix A.) The experiment for Goal 2 has three dependent variables, namely the quality (*Correctness*, *Completeness,* and *Redundancy*) of analysis class diagrams, the quality (*Correctness*, *Completeness*, *Redundancy*, and *Consistency* to the Boundary-Control-Entity principle [Bruegge and Dutoit 2009]) of analysis sequence diagrams, and the correctness of responses to a comprehension questionnaire designed for this experiment.

3.2. Context Selection and Subjects

The context of both experiments is a fourth-year Software Engineering course at Carleton University, Ottawa, Canada. Since this course is the next to the last software engineering course in the curriculum, it is an opportunity for the students to use the skills they have acquired in previous courses regarding requirements elicitation and object-oriented analysis and design. The subjects selected were 34 (first experiment) and 39 (second experiment) students registered in the course in 2008 and 2009, respectively.

A presentation was first provided to the students before each experiment, focusing on the restriction rules and the use case template, and how they fit into the requirements elicitation and specification phase. An assignment was designed to train the students on how to apply the restriction rules and the use case template before the experiments. The result of the assignment was used to group the participants into two blocks and therefore ensure better homogeneity across the two groups involved in the experiment (Section 3.4) for the first experiment. In the second experiment, the participants were grouped into four blocks according to the results of the first two labs of this course, one of which was about identifying inconsistency between diagrams in a UML analysis model and the other was designed for the students to practice metamodeling using the UML class diagram notations.

Two applications, namely a Video Store (VS) system and a Car Parts Dealer (CPD) system are used as experimental materials. These two systems are of similar complexity in terms of the number of use cases in their UCMods, the number of classes in their class diagrams, and the complexity of each UCS. Experimental materials must necessarily be of limited complexity since we have to consider whether the participants are able to finish the prescribed tasks, being described in Section 3.4, within two (Experiment 1) and four (Experiment 2) 3-hour-long laboratory sessions.

The experiments were part of a series of compulsory laboratory exercises that were part of the course curriculum. After the experiments, the participants were asked to sign a consent form to indicate whether they allowed their laboratory results to be used for research purposes. The experiment plan had been reviewed by and received clearance through the Carleton's Research Ethics Committee before collecting the consent forms. Participants who participated in the experiments signed a consent form to confirm their agreement on our using of the collected data for research purposes.

3.3. Hypotheses Formulation

In this section, we only formulate experimental hypotheses for our second research goal as the first one exclusively focuses on characterizing the ease with which developers can apply the restriction rules and does not involve a comparison. The experiment for Goal 2 (Experiment 2) has one independent variable *Method*, with two treatments: *UCM_R* and *UCM_UR*, respectively denoting the use or not of RUCM, and three dependent variables *CD*, *SD*, and *QC*, respectively denoting the quality of analysis class diagrams, the quality of analysis sequence diagrams, and the correctness of responses to a comprehension questionnaire.

Table V. Hypotheses – Experiment 2

Dependent Variable	Null Hypothesis	Alternative Hypothesis
Quality of analysis class diagram	H_0: *CD(UCM_R) = CD(UCM_UR)*	H_1: *CD(UCM_R) ≠ CD(UCM_UR)*
Quality of analysis sequence diagram	H_0: *SD(UCM_R) = SD(UCM_UR)*	H_1: *SD(UCM_R) ≠ SD(UCM_UR)*
Correct response rate of the comprehension questionnaire	H_0: *QC(UCM_R) = QC(UCM_UR)*	H_1: *QC(UCM_R) ≠ QC(UCM_UR)*

Table VI. Participant Groups and Tasks – Experiment 1

Lab	Task	G1		G2		Obtained data points	
		G1.1	G1.2	G2.1	G2.2		
1	Defining UCSs	VS		CPD		26	
2	Deriving analysis models	CPD in UCM_R	CPD in UCM_UR	VS in UCM_UR	VS in UCM_R	UCM_R	UCM_UR
						14	12

Based on the above variables, we can formulate the following null hypothesis (H_0) to be tested for each dependent variable for Goal 2: there is no significant difference between UCM_R (RUCM) and UCM_UR (with the standard template and unrestricted text) in terms of CD, SD, and QC. The alternative hypotheses (H_1) is then two-tailed and is stated as: UCM_R results in different quality of analysis models, or different correctness of responses to the comprehension questionnaire when compared to UCM_UR. Formal hypotheses are provided in Table V.

3.4. Experiment Design

In this section, we report the experiment design of the first experiment in Section 3.4.1 followed by the rationale for conducting a replication (the second experiment) and its experiment design (Section 3.4.2).

3.4.1. Experiment 1. The participants were asked to perform two tasks over two laboratories (3 hours each). Task 1 (defining UCMods) involved writing UCSs by applying RUCM (Section 2). In this task, the use case diagrams of the two systems, partially filled UCSs, and a comprehension questionnaire designed for evaluating the restrictions rules, were provided as input documents to the participants. Task 2 (deriving UML analysis models) involved the construction of analysis models from two types of UCMods, one of which applied RUCM (UCM_R) whereas the other only applied a standard template without restrictions (UCM_UR).

As stated previously, an assignment was designed to train the participants to apply RUCM. The assignment was essentially similar to Task 1 except that a different system was used and the participants were not monitored while they did their assignment. Individual feedbacks were given to each participant and a solution of the assignment was also provided before the experiment was conducted. Based on the grades of this assignment, we defined the following three blocks: grades B to A+ (15 participants), grades B− to F (13 participants), and absent (ABS) (6 participants). As shown in Table VI, the participants were then divided into two groups: A and B. Each of the two groups was then randomly assigned participants from the three blocks in nearly identical proportions.

In Lab 1, the participants in group G1 were asked to complete UCSs of the VS system by applying RUCM, whereas the participants in group G2 did the same task on the CPD system. In Lab 2, we further divided the participants of G1 into groups G1.1 and G1.2, so that the participants in G1.1 could derive class and sequence diagrams from the UCM_R use case model for the CPD system, while the participants in G1.2 did the same from the UCM_UR use case model. The same strategy was followed for group G2.

Table VII. Participant Groups and Tasks – Experiment 2

Lab	Task	G1	G2	G3	G4
1	Deriving class diagram (Task A)	CPD in UCM_R	CPD in UCM_UR	VS in UCM_R	VS in UCM_UR
2	Deriving sequence diagram (Task B)	CPD in UCM_R	CPD in UCM_UR	VS in UCM_R	VS in UCM_UR
3	Deriving class diagram (Task A)	VS in UCM_UR	VS in UCM_R	CPD in UCM_UR	CPD in UCM_R
4	Deriving sequence diagram (Task B)	VS in UCM_UR	VS in UCM_R	CPD in UCM_UR	CPD in UCM_R

When assigning participants to sub-groups we followed the same blocking strategy as the one used to create groups G1 and G2. Note that we used different systems for the two labs for each group of participants to limit learning effects that would otherwise constitute a threat to validity. For example, G1 used 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).

3.4.2. Experiment Replication (Experiment 2). In Lab 2 of Experiment 1, the participants were asked to derive both class and sequence diagrams. However, most of the participants were not able to finish sequence diagrams due to time constraints and therefore we were not able to evaluate the impact of RUCM on the quality of manually created analysis sequence diagrams. Besides, no statistically significant differences were observed for analysis class diagrams in terms of their completeness and redundancy in Experiment 1 and we thought it was perhaps also due to the time constraints. Therefore, in Experiment 2, we replicated Task 2 to 1) evaluate the impact of RUCM on the quality of analysis sequence diagrams and 2) to see whether significant differences between the two treatments can be identified in terms of completeness and redundancy of generated analysis class diagrams if more time is given to participants. Task 2 was split into two tasks in Experiment 2: one was to construct an analysis class diagram (Task A) and the other was to construct analysis sequence diagrams (Task B). The participants in Experiment 2 were asked to perform Task A and Task B over two consecutive laboratories (3 hours each).

Based on the average grades of two laboratories preceding the experiment, we defined the following four blocks: grades A+ (11 participants), grades A and A− (12 participants), grades B+ – C+ (9 participants), and grades C – D− (7 participants). As shown in Table VII, the participants were then divided into four groups: G1, G2, G3, and G4. Each of the four groups was then randomly assigned participants from the four blocks in nearly identical proportions.

In Lab 1, the participants in group G1 were asked to derive analysis class diagrams from the UCMod with restrictions for the CPD system, while the participants in group G2 did the same from the UCMod without restrictions. The same strategy was followed for groups G3 and G4. In Lab 2, the participants were provided the same UCMod as the one they were assigned in Lab 1 but they were asked to derive analysis sequence diagrams and at the same time keep consistency between these sequence diagrams and the class diagrams they designed in Lab 1. In other words, the participants were told to refine their class diagrams designed in Lab 1 whenever necessary. At the end of the lab, the participants were asked to submit their original class diagram from Lab 1, refined class diagram, and sequence diagrams from Lab 2. In Lab 3 and Lab 4, each participant group performed Task A and Task B with a different system and a different treatment. As a result, each group executed different combinations of treatment and system and therefore learning effects that would constitute a threat to validity were limited.

3.5. Instrumentation

The instruments of an experiment are classified into three types: experiment objects, guidelines, and measurement instruments [Wohlin and Wesslen 2000]. In this section we discuss our experiment instruments by conforming to this classification.

3.5.1. Task 1 – Derive Use Case Specifications. Use cases are specified by applying RUCM (Section 2.1). Inputs are the use case diagrams of the two systems, the partially filled UCSs, and a comprehension questionnaire designed to capture the participants' subjective opinions on the restrictions rules. These input documents are further discussed below. The UCSs completed by the participants and their responses to the comprehension questionnaire were collected and used to evaluate the application of each restriction rule.

3.5.1.1. Experiment objects. For each system (CPD and VS), the experiment objects are composed of a use case diagram, a set of UCSs, and a system description. These two systems were originally designed as the lab materials for the Software Engineering course (4th-year undergraduate course) and have been used for several years. Each document contains a textual system description, a use case diagram along with UCSs following a standard template [Bruegge and Dutoit 2009], a class diagram, and sequence diagrams for a subset of the use cases. The use case diagrams and the textual system descriptions are reused as experiment objects without making any changes. Since Task 1 requires that the participants document UCSs by applying RUCM, we provided the participants the partially filled UCSs complying with our use case template. The UCSs were partially filled and contain descriptions for only the following fields of the template (Section 2): *Use Case Name*, *Brief Description*, *Primary Actor*, *Secondary Actor*, *Dependency*, and *Generalization*. The rationale was to provide an overview of high level requirements (e.g., thanks to the brief description), ensure UCSs were consistent with the use case diagram, and let the participants focus on defining flows of events, which is the most complex part of UCSs and on which most of our restriction rules apply.

3.5.1.2. Experiment guidelines. A lab description was provided to the participants at the beginning of each lab, describing the list of documents provided, the task of the lab, and the submission guidelines. The participants were asked to complete five UCSs during the three-hour lab. The lab description also made it clear that the restriction rules had to be applied and this was a very important component of the evaluation of lab results.

3.5.1.3. Measurement instruments. An evaluation questionnaire was designed for the participants to characterize each restriction rule according to three measures: *Understandability*, *Applicability*, and *Restrictiveness*. Three questions or statements were designed to capture these three measures on an appropriate scale (Table VIII). The first question is a "yes/no" question to capture whether the participants perceived they were able to understand each restriction rule and were able to properly apply them at the time when the lab task was performed. The second statement is evaluated on a four-point Likert scale [Oppenheim 1992] question, which requires the participants to rate each restriction rule according to the extent to which they perceive it to be straightforward to apply. The third statement is also defined on a four-point Likert scale. It is used to capture the perceived restrictiveness of each restriction rule. The complete questionnaire is provided in [Yue et al. 2010b] for reference.

3.5.2. Task 2 – Derive Analysis Models. Task 2 consists in deriving analysis models from two types of UCMods: One UCMod documented in RUCM (UCM_R) (e.g., groups A1 and B1 in Experiment 1—Section 3.4) whereas the other only uses a standard

Table VIII. Comprehension Questionnaire of Task 1

Measure	Question/Statement	Scale
Understandability	I understood the restriction rule and was able to properly apply it. (Question asked for each rule.)	Yes / No
Applicability	The restriction rule is straightforward to apply. (Question asked for each rule.)	4-point Likert scale: Completely agree, Generally agree, Generally disagree, Completely disagree
Restrictiveness	The restriction rule was too restrictive. (Question asked for each rule.)	4-point Likert scale: Completely agree, Generally agree, Generally disagree, Completely disagree

template [Bruegge and Dutoit 2009] (UCM_UR) (e.g., groups A2 and B2). Notice that the participants were equally trained to understand our use case template and the standard template. With the provided UCMods, in Experiment 1, the participants were asked to manually derive an analysis model (class and sequence diagrams) and also answer a comprehension questionnaire to assess their understanding of the use cases. In Experiment 2, the participants were asked to derive a class diagram for two different systems in lab 1 and lab 3 (Task A), and derive corresponding sequence diagrams and answer a comprehension questionnaire to assess their understanding of the use cases in lab 2 and lab 4 (Task B).

The standard template [Bruegge and Dutoit 2009] of UCM_UR that we used as a comparison baseline to document UCSs shares common fields with our template (see an example in Appendix A): use case name, brief description, precondition, primary actor, secondary actors, dependency, and generalization. However, it is missing the following characteristics (which are specified as restriction rules in RUCM):

—It does not require that each flow of events (both basic flow and alternative flow) of a UCS contains its own postcondition.
—It does not distinguish different types of alternative flows.
—It does not require that the action steps of a UCS match one of the five interaction patterns we specified in Section 2.1.
—It does not enforce using any keyword to clearly specify interactions between the basic flow and its corresponding alternative flows, contrary to our template.

3.5.2.1. Experiment objects. The CPD and VS systems come in two versions for Task 2: they contain the same use case diagram but have different UCSs (with or without restrictions). As previously mentioned in Section 3.5.1.1, the CPD and VS systems were originally designed as the lab materials for a Software Engineering course (4th year undergraduate course) and were therefore constructed for teaching purposes. They have been used for several years and the original version of the UCMods of CPD and VS was used for treatment UCM_UR in our experiment. T. Yue rewrote all the UCSs applying RUCM and the resulting UCMods were therefore used for treatment UCM_R. Both sets of UCSs were then carefully reviewed by the authors to ensure that they both contain the same information, though in different forms, so that the participants have the possibility of creating identical analysis models from them. An example use case documented with and without RUCM is given in Appendix A.

3.5.2.2. Experiment guidelines. As for Task 1 (Section 3.5.1.2), in Task 2, we also provided the participants a lab description at the beginning of the lab, describing the list of documents provided, the task of the lab, and the submission guidelines. Two examples of such guidelines are provided in Appendix B for reference. With the UCMods as input documents, the participants were asked to design a class diagram. We made it clear in the lab description (Experiment 1: lab 2, Experiment 2: lab 1 and lab 3) that the participants should, based on the use case descriptions, assign meaningful names for each class, attribute, and operation, and apply the well-established *Entity/Boundary/Control* stereotype classification [Bruegge and Dutoit 2009] for each class. The participants were also asked to complete a comprehension questionnaire during the lab (Experiment 1: lab 2, Experiment 2: lab 2 and lab 4), which was designed to evaluate how well they were able to understand the flow of events of each UCS. In Experiment 1, lab 2, the participants were also asked to derive sequence diagrams for two selected use cases; however most of the participants were not able to derive (complete) diagrams due to time constraints, and we therefore decided not to analyze them. In Experiment 2, deriving sequence diagrams (Task B) was completed in lab 2 and lab 4. The participants working on the CPD (VS) system were asked to derive three (two) sequence diagrams for three (two) selected use cases. Participants worked on different numbers of use cases to ensure balanced tasks given the differences of the complexity of UCSs: the three CPD use cases had a comparable, overall complexity to that of the two VS use cases. We measure the complexity of a use case by simply calculating the total number of condition and action sentences they contain. Note that all the participants were trained with the same methodology for deriving a UML analysis model from a use case model, as part of the Software Engineering course that the participants were taking. This methodology was proposed by Bruegge and Dutoit [2009].

3.5.2.3. Measurement instruments. A comprehension questionnaire was designed for each system to quickly evaluate, in a repeatable and unbiased way, the extent to which a participant understood the main body (flows of events) of each UCS. To avoid introducing any bias, we ensured comprehension questions were answerable by both the participants using the restricted and unrestricted UCMods.

Questions in the comprehension questionnaires are grouped per use case so that the questionnaires can be easily browsed. The complete questionnaires for the two systems are provided in Yue et al. [2010b]. Each question covers either one of the following aspects of UCSs: action sequences, object responsibilities, conditions triggering actions, and general comprehension. An example question for each aspect is provided in Yue et al. [2010b] for reference. Each multiple-choice question includes two choices, namely "Not specified in the UCS" and "Other answer," so that questions are at the same time closed and open, thereby allowing us to collect complete information. (For instance, some ambiguities in UCSs may prevent the participants from selecting one of the available choices, in which case they can use either one of these two specific choices.) For each question, the participants were also asked to indicate where they found relevant information to answer the question (location). These two special choices, along with the location question, should also significantly reduce the chances that participants randomly select a choice. During data collection, we checked consistency between the selected choice and the location answer: no inconsistency was observed. For each use case, a general comprehension question was asked to determine whether the participants identified any ambiguity in the corresponding UCS. The participants' responses to this question were carefully verified to determine the validity of the data collected, but were not directly used to test our experimental hypotheses. (Section 3.3). In other words, it is not counted for measuring the correctness of responses to the comprehension questionnaires (Section 3.6.5).

All multiple-choice questions contain one and only one correct answer, except for one question in the comprehension questionnaire of the VS system, where two choices are partially correct and together make up a complete, correct answer to the question. Therefore, when grading answers to this question we considered both choices as correct and we also considered "other" as correct if the correct explanation were provided. If a response to a multiple-choice question was "Not specified in the UCS," we assumed

Table IX. Measures Used to Derive Error Rate

Measure	Specification (a single UCS)
N_{v_r}	# of violations of a specific restriction rule *r*
N_{missed_r}	# of instances where the keyword (defined as the restriction rule *r*) should be applied, but is not
N_{ASteps}	Total number of action steps (sentences)
N_{Cond}	Total number of condition sentences: precondition, postcondition, IF/ELSEIF condition, VALIDATES THAT condition, and UNTIL condition
$N_{Include}$	Total number of INCLUDE USE CASE sentences
N_{Extend}	Total number of EXTENDED BY USE CASE sentences
$N_{AltFlows}$	Total number of alternative flows
N_{NP}	Total number of noun phrases
N_{IF}	Total number of IF-THEN-ELSE-ELSEIF-ENDIF conditional logic sentences
N_{ABORT}	Total number of ABORT sentences
N_{RESUME}	Total number of RESUME STEP # sentences
$N_{MEANWHILE}$	Total number of MEANWHILE sentences
N_{VALTS}	Total number of VALIDATES THAT sentences
$N_{DO\text{-}UNTIL}$	Total number of DO-UNTIL sentences
N_{RFS}	Total number of the places where the keyword RFS is applied
N_{Spe}	$=N_{ABORT}+N_{RESUME}$
N_{Dep}	$=N_{Include}+N_{Extend}$
$N_{NormalA}$	$=N_{ASteps}-N_{Spe}-N_{Dep}$
$N_{NormalS}$	$=N_{NormalA}+N_{Cond}$
N_{AllS}	$=N_{ASteps}+N_{Cond}-N_{VALTS}$

that there either exists an ambiguity in the UCS, which further leads to a low quality analysis model, or the participants were not able to correctly answer the question. Such an answer was therefore considered to be incorrect. If the response to a multiple-choice question is “Other,” then the response was carefully checked to see whether it was equivalent to the correct answer.

Questionnaire characteristics for the two systems are summarized in Yue et al. [2010b]. In total, fifteen questions contribute to the measurement of the comprehension questionnaire for the CPD system; while twenty-five questions contribute to that of the VS system. The difference in the numbers of the questions is simply due to the different numbers of UCSs in the two systems.

3.6. Evaluation Measurement and Data Collection

The goal of Task 1 is to evaluate each restriction rule based on four measures: *Error Rate*, *Understandability*, *Applicability*, and *Restrictiveness*. The goal of Task 2 is to evaluate the impact of either our restriction rules and template or a standard template on the quality of manually derived analysis models, with three dependent variables: the quality of class diagrams (abbreviated as CD), sequence diagrams (abbreviated as SD), and the correctness of responses to the comprehension questionnaires (abbreviated as QC). All these measures and their data collection are described next.

3.6.1. Error Rate. The *Error Rate* of a restriction rule represents the rate of improper applications of the rule, on the scale from 0 to 1. Error rates are presented in Table X (first 16 rules) and Table XI (last 10 rules), and are based on raw data as measured according to metrics in Table IX. The first column of Table X and Table XI lists the restriction rule number. The second column describes the formulas used to calculate the *Error Rate* of each rule. In Table IX, the first column provides the names of the metrics and the second column specifies the metrics or their calculations. The variable

Table X. Error Rate Measurements (R1–R16)

#	Measure	#	Measure	#	Measure	#	Measure
R1	$N_{v\_1}/N_{NormalA}$	R5	$N_{v\_5}/N_{NormalA}$	R9	N_{v_9}/N_{NP}	R13	$N_{v\_13}/N_{NormalS}$
R2	N_{v_2}/N_{ASteps}	R6	$N_{v\_6}/N_{NormalA}$	R10	$N_{v\_10}/N_{NormalS}$	R14	N_{v_14}/N_{NP}
R3	$N_{v\_3}/N_{NormalA}$	R7	$N_{v\_7}/N_{NormalA}$	R11	$N_{v\_11}/N_{NormalS}$	R15	$N_{v\_15}/N_{NormalS}$
R4	N_{v_4}/N_{ASteps}	R8	$N_{v\_8}/N_{NormalS}$	R12	$N_{v\_12}/N_{NormalS}$	R16	N_{v_16}/N_{NP}

Table XI. Error Rate Measurements (R17–R26)

#	Measure	Description (N_{m_r})—see Table IV for the corresponding rules/keywords
R17	$(N_{missed\_17}/N_{Alls} + N_{m\_17}/N_{Include})/2$	# of instances where the keyword is applied, but either in a wrong situation, without the included use case name, or with incorrect use case name, or should not be applied at all.
R18	$(N_{missed\_18}/N_{AllS} + N_{m\_18}/N_{Extend})/2$	# of instances where the keyword is applied, but either in a wrong situation, without the extended use case name, or with incorrect use case name.
R19	$(N_{missed\_19}/N_{AltFlows} + N_{m\_19}/N_{RFS})/2$	# of instances where the keyword is applied, but either in a wrong situation, without the basic flow step number followed, or with a wrong basic flow step number.
R20	N_{m_20}/N_{IF}	# of instances where the keyword is incorrectly applied, such as incorrect grammar.
R21	$(N_{missed\_21}/N_{AllS} + N_{m\_21}/N_{MEANWHILE})/2$	# of instances where the keyword is applied, but in a wrong situation, or the action steps connected by the keyword are not appropriate (e.g., non-action steps).
R22	N_{m_22}/N_{VALTS}	# of instances where the alternative case is not described in its corresponding alternative flow, the condition sentence is not a complete sentence, or there is no condition sentence followed by the keyword.
R23	$(N_{missed\_23}/N_{AllS} + N_{m\_23}/N_{DO\text{-}UNTIL})/2$	# of instances where the keyword is applied, but in a wrong situation, or the grammar is not followed: e.g., missing condition.
R24	$(N_{missed\_24}/N_{AllS} + N_{m\_24}/N_{ABORT})/2$	# of instances where the keyword is applied in a wrong place: e.g., not the last step of an alternative flow.
R25	$(N_{missed\_25}/N_{AllS} + N_{m\_25}/N_{RESUME})/2$	# of instances where the keyword is applied, but not in an alternative flow, or the keyword is applied in an alternative flow, but not in the last step of an alternative flow.
R26	$N_{v\_26}/(N_{AltFlows}+1)$	$N_{AltFlows}$**+1**: # of the flows of events of a UCS.

N_{m_r} specific to the formulas (Table XI, Column 2) for rule *r* (one of the rules R17–R25) is defined in Table XI, Column 3.

The *Error Rate* of each restriction rule equals (Table X) the number of violations of the rule *r* (N_{v_r}) divided by the total number of steps where the rule could be applied, that is, either $N_{NormalA}$, N_{ASteps}, N_{NP}, or $N_{NormalS}$. For example, R1 is specified as "The subjects of sentences in basic and alternative flows should be the system or actors." It puts restriction on action steps, except those describing use case relationships (i.e., those that use keywords INCLUDE USE CASE and EXTENDED BY USE CASE) and those specifying what happens at the end of an alternative flow (i.e., those that use keywords ABORT and RESUME). Therefore the error rate of R1 for a UCS equals $N_{v\_1}/N_{NormalA}$, where N_{v_1} is the number of violations of R1 in the UCS and $N_{NormalA}$ (Table IX) is the total number of action steps (N_{ASteps}) of the UCS that do not contain keywords ABORT, RESUME, INCLUDE USE CASE and EXTENDED BY USE CASE (we substract N_{Spe} and N_{Dep} from N_{ASteps} in Table IX). Some restriction rules require two error rate measures. For example (Table XI), the error rate of R17 is calculated as the average of N_{missed_17}/N_{Alls} and $N_{m\_17}/N_{Include}$, where N_{missed_17}/N_{Alls} captures the rate of obsence of keyword INCLUDE USE CASE (specified in R17), i.e., it should be used but it is not, whereas $N_{m\_17}/N_{Include}$ captures the rate of erroneous occurrences

Table XII. Measures Used to Derive CD

#	Measure	Specification
1	N_{class}	# of classes in the reference model
2	N_{asso}	# of associations in the reference model
3	N_{gen}	# of generalizations in the reference model
4	N_{attr}	# of attributes in entity classes in the reference model
5	N_{opr}	# of operations of control or boundary classes in the reference model
6	N_{cp1}	# of missing attributes of a class
7	N_{cp2}	# of missing operations of a class
8	N_{clp1}	# of missing classes in a class diagram
9	N_{clp2}	# of missing associations in a class diagram
10	N_{clp3}	# of missing generalizations in a class diagram
11	N_{c}	# of matching classes in a class diagram
12	N_{eclass}	# of classes in a class diagram
13	N_{clr2}	# of incorrect associations of a class diagram
14	N_{a}	# of matching associations between matching class diagrams
15	N_{re}	# of extra classes that are redundant, excluding equivalent model elements

(the keyword is used, but either in a wrong situation, without the included use case name, or with an incorrect use case name, or should not be applied at all).

For each UCS, the error rate of a specific restriction rule r ($ErrorRate_r$) is:

$$ErrorRate_r = \sum_{s=1}^{S}\left(\sum_{u_s=1}^{U_s} ErrorRate_{us}\right)\Big/ US,$$

where S is the set of participants (s is an index identifying each participant), U_s is the set of UCSs created by participant s (u_s is an index identifying UCS u created by participant s), US denotes the total number of UCSs written by all the participant. $ErrorRate_{us}$ denotes the error rate of the UCS u created by participants s.

3.6.2. Understandability, Applicability and Restrictiveness. *Understandability* is one of the three subjective measures used to assess the restriction rules and is based on responses to the first (yes/no) question of the comprehension questionnaire of Task 1 (Section 3.5.1.3). This measure, normalized on the scale from 0 to 1, is the ratio of "yes" in all collected responses. The other two subjective measures, *Applicability* and *Restrictiveness*, are measured by using the participant responses to the second and third questions of the comprehension questionnaire of Task 1 (Section 3.5.1.3), respectively. Recall that both questions are answered on four-point Likert scales, from 1 (Completely disagree) to 4 (Completely agree). Our analysis will focus on comparing average scores across all participant responses.

3.6.3. Quality of Analysis Class Diagram (CD). The quality of an analysis class diagram is evaluated from three aspects: *Correctness*, *Completeness*, and *Redundancy*. The reference class diagrams, used as the basis to evaluate class diagrams designed by the participants, are the original class diagrams of the two case study systems we used in Task 2 (Section 3.5.1.1). Data are collected for the first five measures in Table XII from the reference class diagrams (e.g., number of classes in each reference class diagram (N_{class})); while data are also collected for the last 10 measures in Table XII from the class diagram of each participant. All these data are then used to compute the measures of *Completeness*, *Correctness* and *Redundancy* of a participant class diagram according to the formulas of Table XIII. The completeness of a class diagram ($R_{CDcomplt}$) is computed as the average of Class Completeness (R_{clp1}), Association Completeness

Table XIII. Quality Measures for a Class Diagram

Category	Measure	Formula
Completeness ($R_{CDcomplt}$)	Class Completeness $R_{clp}1 = 1 - N_{clp1}/N_{class}$	$R_{CDcomplt} = (R_{clp1} + R_{clp2} + R_{clp3})/3$
	Association Completeness $R_{clp2} = 1 - N_{clp2}/N_{asso}$	
	Generalization Completeness $R_{clp3} = 1 - N_{clp3}/N_{gen}$	
Correctness ($R_{CDcorrect}$)	Class Correctness $R_{clr}1 = 1 - \sum_{i=1}^{N_c} R_{ci} \Big/ N_c$	$R_{CDcorrect} = (R_{clr1} + R_{clr2})/2$
	Association Correctness $R_{clr2} = N_{clr2}/N_a$	
Redundancy (R_{CDre})	$R_{CDre} = N_{re}/N_{eclass}$	

(R_{clp2}) and Generalization Completeness (R_{clp3}) since we consider classes, associations and generalizations to be the three most important (structural) aspects of a class diagram. The class diagram correctness ($R_{CDcorrect}$) is determined by the correctness of classes (R_{Ci})—computed as the average, over the complete class diagram, of the class measures of *Completeness* ($R_{Ccomplt}$) and *Correctness* ($R_{Ccorrect}$) (Table XIV)—and associations. The class diagram redundancy (R_{CDre}) is computed as the ratio of redundant classes (N_{re}) over all the classes of a participant's class diagram (N_{eclass}).

For each class of the reference class diagram of a case study system, we look for a class with the same name in a participant class diagram. If such a matching class is found, then it is evaluated according to the quality measures for a class (Table XIV); otherwise, we keep looking for a design equivalent[1] to the reference class in the participant class diagram. If no such equivalent design exists, then we consider the reference class as missing and therefore the participant diagram as incomplete. When all the reference classes have been looked at, we obtain three outputs: 1) A set of matching classes which are evaluated by using the quality measures for a class (Table XIV); 2) A set of equivalent designs, which are not measured because this would require either a subjective measurement or a large number of specific measures. Besides not many such equivalent designs have been found and not measuring them does not really impact the measurement of CD; 3) A set of reference classes, missing in the participant class diagram. A procedure similar to this identification of matching classes, missing classes, and equivalent class designs is applied to identify matching/missing/equivalent attributes, operations, associations, and generalizations. Each participant class diagram is evaluated by applying the quality measures for a class diagram (Table XIV).

3.6.4. Quality of Analysis Sequence Diagram (SD). Five measures evaluate the quality of a sequence diagram (SD): SD *Completeness*, SD *Correctness*, SD *Redundancy*, and *BCE Consistency* (consistency with the Boundary/Control/Entity principle). They are defined (Table XVI) based on simpler measures (Table XV). To evaluate the quality of sequence diagrams produced by participants we need reference sequence diagrams. Unfortunately we do not find sequence diagrams for the CPD and VS systems in reference textbooks or other resources. We only possess sequence diagrams manually created by the authors of this article. However, we can use sequence diagrams generated by the CASE tool aToucan, which automatically generates analysis class and sequence diagrams from RUCM requirements, since these diagrams have been shown to be of

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

Table XIV. Quality Measures for a Class

Category	Measure	Formula
Completeness ($R_{Ccomplt}$)	Missing stereotype (R_{cp1})	Entity classes: $R_{Ccomplt} = 1 - (R_{cp1} + R_{cp2})/2$ Boundary classes: $R_{Ccomplt} = 1 - (R_{cp1} + R_{cp3})/2$
	Missing attributes $R_{cp2} = N_{cp1}/N_{attr}$	
	Missing operations $R_{cp3} = N_{cp2}/N_{opr}$	
Correctness ($R_{Ccorrect}$)	Incorrectly named (R_{cr1})	Entity classes: $R_{Ccorrect} = 1-(R_{cr1}+R_{cr2}+R_{cr3}+R_{cr4}+R_{cr5}+R_{cr6})/6$ Boundary classes: $R_{Ccorrect} = 1 - (R_{cr1} + R_{cr2} + R_{cr3} + R_{cr5} + R_{cr6})/5$ If the entity class under evaluation does not contain any attributes, then: $R_{Ccorrect} = 1 - (R_{cr1} + R_{cr2} + R_{cr3} + R_{cr5})/4$ If the boundary class under evaluation does not contain any operations, then: $R_{Ccorrect} = 1 - (R_{cr1} + R_{cr2} + R_{cr3} + R_{cr5})/4$
	Incorrectly stereotyped (R_{cr2})	
	Incorrectly assigned "abstract" (R_{cr3})	
	Does not represent one and only one logical concept (R_{cr4})	
	Not given a cohesive set of responsibilities (R_{cr5})	
	Does not represent the intended meaning of the class (R_{cr6})	
Class i (R_{Ci})	$R_{Ci} = (R_{Ccomplt} + R_{Ccorrect})/2$	

Table XV. Measures Used to Derive SD

#	Measure	Specification
1	$N_{msg\text{-}r}$	# of messages in the reference sequence diagram
2	$N_{lifeline\text{-}r}$	# of lifelines in the reference sequence diagram
3	$N_{IU\text{-}r}$	# of interaction uses in the reference sequence diagram
4	$N_{CF\text{-}r}$	# of combined fragments in the reference sequence diagram
5	$N_{msg\text{-}t}$	# of messages in a tested sequence diagram
6	$N_{lifeline\text{-}t}$	# of lifelines in a tested sequence diagram
7	$N_{IU\text{-}t}$	# of interaction uses in a tested sequence diagram
8	$N_{CF\text{-}t}$	# of combined fragments in a tested (participant) sequence diagram
9	$N_{mMsg\text{-}t}$	# of missing messages in a tested sequence diagram
10	$N_{mIU\text{-}t}$	# of missing interaction uses in a tested sequence diagram
11	$N_{mCF\text{-}t}$	# of missing combined fragments in a tested sequence diagram
12	$N_{iMsg\text{-}t}$	# of occurrences of incorrect messages (i.e., incorrect message name, incorrect lifelines, wrong direction, and/or incorrect message types) in a tested sequence diagram
13	$N_{iIU\text{-}t}$	# of occurrences of incorrectly applied interaction uses in a tested sequence diagram
14	$N_{iCF\text{-}t}$	# of occurrences of incorrectly applied combined fragments 1. InteractionOperand is not provided or a wrong operand is used. 2. Condition is not provided or a wrong condition is used. 3. Lifelines are not correctly covered. 4. Wrong messages are covered.
15	$N_{iSeq\text{-}t}$	# of occurrences of incorrect sequences of messages, interaction uses, or combined fragments.
16	$N_{vBCE\text{-}t}$	# of violations of the Boundary-Control-Entity (BCE) principle
17	$N_{reMsg\text{-}t}$	# of extra messages that are redundant; the semantics of these messages have been modeled by other elements

high quality [Yue et al. 2010b]. Data are collected from the aToucan reference sequence diagrams to compute the first four measures in Table XV, while data are collected from the sequence diagrams being evaluated for the last 12 measures.

As shown in Table XVI, the *completeness* of a sequence diagram ($R_{SDcomplt}$) is computed as the average of the *Message Completeness* (R_{conMsg}), *IU* (`InteractionUse`) *Completeness* (R_{conIU}), and *CF* (`CombinedFragment`) *Completeness* (R_{conCF}). The consistency of a sequence diagram ($R_{SDcorrect}$) is determined by the consistency of messages (R_{corMsg}),

Table XVI. Quality Measures for a Sequence Diagram

Category	Measure	Formula
Completeness ($R_{SDcomplt}$)	Message Completeness $R_{conMsg} = 1 - \frac{N_{mMsg-t}}{N_{msg-r} - N_{uMsg-r} - N_{aaMsg-r}}$	$R_{SDcomplt} = \frac{R_{conMsg} + R_{conIU} + R_{conCF}}{3}$
	IU Completeness $R_{conIU} = 1 - N_{mIU-t}/(N_{IU-r})$	
	CF Completeness $R_{conCF} = 1 - N_{mCF-t}/(N_{CF-r})$	
Correctness ($R_{SDcorrect}$)	Message Correctness $R_{CorMSg} = 1 - \frac{N_{imsg-t} + N_{aaMsg-t}}{N_{msg-t}}$	$R_{SDcorrect} = \frac{R_{corMsg} + R_{corIU} + R_{corCF} + R_{corSeq}}{4}$
	IU Correctness $R_{corIU} = 1 - N_{iUI-t}/(N_{IU-t})$	
	CF Correctness $R_{corCF} = 1 - N_{iCF-t}/(N_{CF-t})$	
	Message Sequence Correctness $R_{corSeq} = 1 - N_{iseq-t}/N_{msg-t}$	
Redundancy (R_{SDre})	$R_{SDre} = N_{reMsg-t}/N_{msg-t}$	
BCE Consistency (R_{conBCE})	$R_{conBCE} = 1 - N_{vBCE-t}/N_{mMsg-t}$	

interaction uses (R_{corIU}), combined fragments (R_{corCF}), and message ordering (R_{corSeq}) since these are considered to be important aspects in a sequence diagram. The redundancy of a sequence diagram (R_{SDre}) is computed as the ratio of redundant messages ($N_{reMsg-t}$) over all the messages of a student's sequence diagram (N_{msg-t}). Measure *BCE Consistency* (R_{conBCE}) evaluates whether a student's sequence diagram conforms to the Boundary-Control-Entity (BCE) design principle of analysis sequence diagrams [Bruegge and Dutoit 2009]. Any violation of the following three rules is considered to be a violation of the BCE principle and therefore contributes to measure N_{vBCE-t} (Table XV):

—A message is sent from a Boundary object to an Entity object.
—A message is sent from an Entity object to a Control object.
—A message is sent from an Entity object to a Boundary object.

3.6.5. Correctness of Responses to the Comprehension Questionnaires (QC). As discussed in Section 3.5.2.3, data about the correctness of responses to the questions of the comprehension questionnaires of Task 2 (QC), are used to evaluate the understandability of UCSs, which is normalized between 0 and 1:

For the CPD system: $QC_{-CPD} = \textit{number of correct responses} / 15$
For the VS system: $QC_{-VS} = \textit{number of correct responses} / 25$

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

4. EXPERIMENT RESULTS AND ANALYSIS

Recall that our first goal (Section 3.1) is to assess each restriction rule with respect to four measures: *Error Rate*, *Understandability*, *Applicability*, and *Restrictiveness*, whereas the second goal (Section 3.1) is to evaluate the impact of the restriction rules and use case template on the quality of manually derived analysis models, with three dependent variables: *CD*, *SD*, and *QC*, respectively denoting the quality of analysis class and sequence diagrams manually generated by participants and the correctness of participants responses to a comprehension questionnaire. In Table XVII, we clarify the

Table XVII. Mapping between Measurement and Analysis

Task	Experiment	Measurement	Specification section	Result and analysis section
1	1	*Error Rate*	3.6.1	4.1.1
		Understandability	3.6.2	4.1.2
		Applicability	3.6.2	4.1.3
		Restrictiveness	3.6.2	4.1.4
2	1, 2	*CD*	3.6.3	4.2.1
	2	*SD*	3.6.4	4.2.2
	1, 2	*QC*	3.6.5	4.2.3

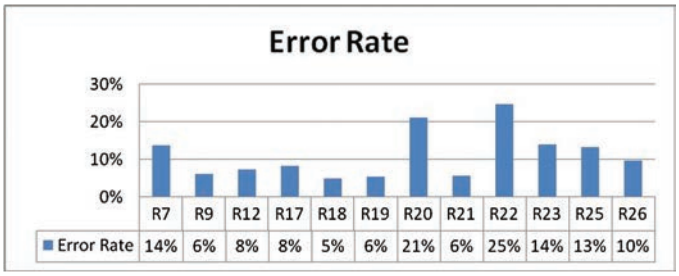
Table XVIII. Collected and Analyzed Data Points – Experiment 2

	Collected data points				Analyzed data points			
Lab	G1	G2	G3	G4	G1	G2	G3	G4
1	10	10	9	10	8	9	9	7
2	10	10	9	8	8	7	8	6
3	10	9	9	10	9	9	8	6
4	9	9	10	8	8	8	9	6

mapping between the subsections of Section 3.1 specifying all the dependent variables and the subsections of Section 3.6 where experiment results and analysis are discussed. In the table, we also indicate which task and experiment relate to which measurement.

In Experiment 1, among the collected 26 data points for Task 2 (Table VI), one participant's result was not included in the analysis because it was very incomplete (no class diagram was derived), thus suggesting he had not complied with instructions. One participant missed the lab for Task 2 and performed the task at home in an uncontrolled manner. We excluded this data point as well. We also excluded a data point from a participant who spent significantly more than the planned three hours (3 hours 40 mins) to finish Task 2. As a result, a total of 23 data points were used for analyzing Task 2 (14 data points for treatment UCM_R and 9 for treatment UCM_UR). Though the two systems used for the experiment may 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.

The collected and used data points in Experiment 2 are provided in Table XVIII. As it is often the case in controlled experiments, unforeseen, difficult to control events lead to the exclusion of invalid data. Some observations in each lab and group were left out of the analysis because of one of the following reasons. 1) Three participants (one in G2 and two in G4) did not want to sign the consent form and therefore their data points were excluded; 2) One participant's result (Lab 2-G1) was very incomplete (no meaningful messages were derived) suggesting he did not follow instructions; 3) Seven participants' results (one in Lab 2-G1, two in Lab 2-G2, one in Lab 2-G3, one in Lab 2-G4, one in Lab 4-G1, and one in Lab 4-G2) contain only one or two sequence diagrams (two or three sequence diagrams are required); 4) Four participants (two in Lab 1-G1, one in Lab 1-G4, and two in Lab 3-G4) spent significantly less than the planned three hours (less than 2 hours) to finish Task A; 5) Two participants in G1 and G4 missed Lab 3 and Lab 2, respectively, and performed the task at home in an uncontrolled manner; 6) One participant (Lab 3-G3) derived a class diagram which is extremely similar to the reference class diagram (e.g., identical class, attribute, and operation names); 7) One participant (G3) derived a single sequence diagram for three use cases of the CPD system in Lab 4.



	R7	R9	R12	R17	R18	R19	R20	R21	R22	R23	R25	R26
■ Error Rate	14%	6%	8%	8%	5%	6%	21%	6%	25%	14%	13%	10%

Fig. 1. Error rates of restriction rules (when ≥ 5%).

We present the experiment results and analysis of the usage of restrictions rules in Section 4.1 and the quality of analysis models in Section 4.2. In Section 4.3, we provide a summary of all the results.

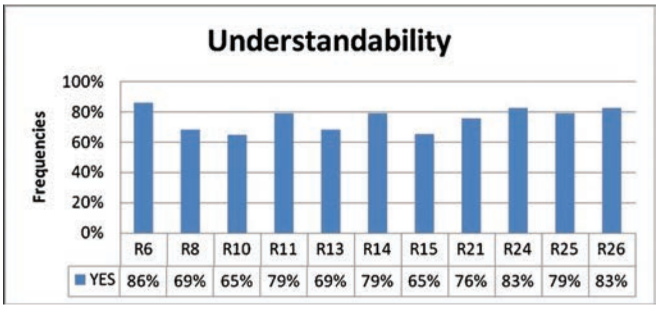
4.1. Usage of Restriction Rules

Each restriction rule is evaluated by four measures: *Error Rate*, *Understandability*, *Applicability*, and *Restrictiveness* (Section 3.6). The last three measure participants' subjective opinions on rules; while *Error Rate* objectively evaluates errors made by participants when applying each rule to UCSs during Experiment 1, Task 1. In this section, experiment results are discussed in terms of each measure.

4.1.1. Error Rate. Twelve of the 26 restriction rules have an error rate above or equal to 5% (Figure 1). Six other rules have nearly no errors. All 26 error rates are provided in Yue et al. [2010b] for reference. Notice that the highest error rate is 25% (R22), and that all but two rules (R20 and R22) have an error rate under 15%. If we now look more closely at R20 and R22, we notice that R20 and R22 are more complex than the others (Section 2.2). R20 contains three different cases, which specify how keywords are used to describe conditional logic sentences (IF-THEN-ELSE-ELSEIF-ENDIF) within a flow or across a reference flow and an alternative flow (e.g., IF-THEN appears in the reference flow while ELSE-ENDIF appears in the alternative flow). We observed that the most frequently occurring errors regarding R20 involve not applying or incorrectly applying the required keywords (e.g., missing ELSE in alternative flow). R22 is also a composite and complex restriction rule, which not only specifies the usage of keyword VALIDATES THAT, but also indicates that the alternative case of the condition checking sentence containing this keyword must be described in an alternative flow.

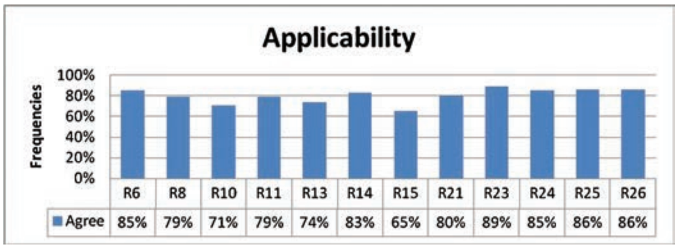
From a more general standpoint, we can also observe from Figure 1 that most of the restrictions on the use of natural language (R1–R16) have error rates lower than the restrictions on the use of control structures specified as keywords (R17–R25). Appropriate tool support (part of our future work) can very likely be used to enforce the proper usage of keywords specified in the latter set of rules, thus potentially reducing their error rates. We also believe that more extensive training, perhaps specifically focused on error-prone rules, could further reduce error rates.

4.1.2. Understandability. The *Understandability* of a restriction rule reflects the ease of understanding the rule according to the subjective opinions of the participants. Figure 2 presents the 11 (out of 26) rules that score below 90%. All 26 *Understandability* scores are provided in Yue et al. [2010b] for reference. The lowest score is 65% (R10 and R15). It is worth mentioning that R8, R10, R13, and R15 rely on concepts from the natural language domain (e.g., "declarative sentence," "modal verb," "negatively adverb," and "participle phrase"), which could probably explain why these four restriction



	R6	R8	R10	R11	R13	R14	R15	R21	R24	R25	R26
■ YES	86%	69%	65%	79%	69%	79%	65%	76%	83%	79%	83%

Fig. 2. Understandability of the restriction rules (when < 90%).



	R6	R8	R10	R11	R13	R14	R15	R21	R23	R24	R25	R26
■ Agree	85%	79%	71%	79%	74%	83%	65%	80%	89%	85%	86%	86%

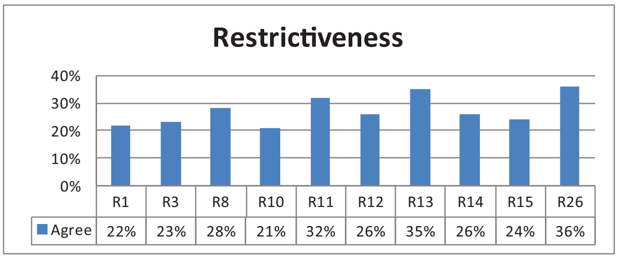
Fig. 3. High applicability of the restriction rules (when < 90%).

rules have relatively low *Understandability* (from 65% to 75%). It is our experience that computer/software engineering participants in Canada have in general a limited understanding of grammatical and natural language concepts, an issue that may also extend to many IT professionals. This suggests that in the future we probably need to put more efforts on these rules during training. Another possible explanation could be that the motivation and rationale of these restriction rules is not clear to the participants since some of them (e.g., R12–R15) are designed to facilitate automated natural language processing, an aspect of the study that the participants were not informed about.

In general, we can see that the participants have received sufficient training before the experiment so that they were able to understand most of the restriction rules to a sufficient degree. Therefore, we are confident that the participants who were given UCMods with restrictions in Task 2 are capable to understand the UCMods and derive analysis models from them.

4.1.3. Applicability. *Applicability* is measured on a four-point Likert scale, from 1 (Completely disagree) to 4 (Completely agree). The frequencies of the participants' responses on this scale are analyzed. Figure 3 presents the percentage of responses with "agree" scores on *Applicability*, showing only rules with a percentage below 90% (12 out of 26 restriction rules). The scores for the 26 restriction rules are provided in Yue et al. [2010b] for reference.

Figure 3 shows that most of the participants (80% and above) agree that most of the restriction rules (21 rules) are easy to apply, except for rules R8, R10, R11, R13 and R15, which receive less than 80% "agree" responses. Notice that these rules also receive relatively low *Understandability* scores (Figure 2). This probably suggests that these rules are relatively difficult to understand and, as a result, they are also relatively hard to apply. As we have discussed in Section 4.1.2, better training and/or tool support could help improve these scores.



	R1	R3	R8	R10	R11	R12	R13	R14	R15	R26
■ Agree	22%	23%	28%	21%	32%	26%	35%	26%	24%	36%

Fig. 4. High restrictiveness of the restriction rules (when $> 20\%$).

4.1.4. Restrictiveness. Restrictiveness is also measured on a four-point Likert scale from 1 (Completely disagree) to 4 (Completely agree). Figure 4 presents the percentage of responses with "agree" scores on *Restrictiveness*, showing only the 10 (out of 26) rules receiving scores above 20%. The scores for all the 26 restriction rules are provided in Yue et al. [2010b] for reference. As shown in Figure 4, most of the participants (80% and above) agree that more than half of the restriction rules (16) are not restrictive. Though the 10 rules shown in Figure 4 all received a score above 20%, all scores are below 40% and the remaining 16 rules were below this threshold. This is not bad considering that the participants applied the restriction rules for the first time and were not informed of the rationale of the restriction rules when the experiment was conducted. In other words, if the participants knew what the rules were for, we conjecture they would probably consider them less restrictive.

4.2. Quality of Analysis Models

As we have discussed in Section 3.3, Goal 2 involves one independent variable (*Method*) with two treatments, *UCM_R* and *UCM_UR*, respectively denoting the use or not of RUCM, and three dependent variables *CD*, *SD*, and *QC*, respectively denoting the quality of analysis class diagrams, the quality of analysis sequence diagrams, and the correctness of responses to a comprehension questionnaire (Sections 3.6.3 and 3.6.4). In this section, we discuss the impact of the independent variable *Method* on the dependent variables *CD* (Section 4.2.1), *SD* (Section 4.2.2), and *QC* (Section 4.2.3) and also investigate the possible interactions between *Method* and two factors (*System* and *Order*) on these three dependent variables.

4.2.1. Quality of Analysis Class Diagram (CD). This section discusses the impact of *Method* (two treatments: *UCM_R* vs. *UCM-UR*) on *CD* (the quality of analysis class diagrams) in Section 4.2.1.1, which is determined by several measures (Section 3.6.3). Next, we investigate possible interactions between *Method* and a number of factors on *CD*: *System* (*CPD* vs. *VS*) and *Order* (2.1[2] vs. 2.3) in Section 4.2.1.2. Detailed descriptive statistics are provided in Yue et al. [2010b] for reference.

One first observation is that all means for *Association Completeness*, for all experiments, are below 0.25. This indicates that less than 25% of the associations in the reference class diagrams were derived by the participants. (Recall that the reference class diagrams of CPD and VS contain 15 and 10 associations, respectively.) Furthermore, many participants were not even able to derive any of the reference associations. In such situations, *Association Correctness*, which is defined as the ratio of correctly identified associations to all identified associations and also contributes to the

[2]In the rest of the article, we use 2.1, 2.2, 2.3, and 2.4 to denote the first, second, third, and fourth labs of Experiment 2, respectively.

Table XIX. Two Tailed t-test and Wilcoxon Test – CD

			t-test				Wilcoxon test
Exp	System	Measures	Mean difference (UCM_R – UCM_UR)	DF	t-value	p-value	Prob > |Z|
1	CPD+VS	Class Completeness	0.108	15	1.334	0.2022	0.1848
		CD Completeness	0.082	17	1.552	0.139	0.1564
		Class Correctness	0.138	16	**2.281**	**0.037**	**0.0298**
		Redundancy	−0.048	12	−0.792	0.4436	0.5372
2.1	CPD	Class Completeness	0.213	12	**2.827**	**0.0157**	**0.0203**
		CD Completeness	0.124	11	1.813	0.0961	0.1356
		Class Correctness	0.058	13	1.757	0.1022	0.1937
		Redundancy	−0.105	14	**−2.269**	**0.0398**	0.1472
	VS	Class Completeness	0.051	12	0.99	0.3411	0.4775
		CD Completeness	0.076	10	1.384	0.1956	0.39
		Class Correctness	0.09	7	1.986	0.0875	0.0567
		Redundancy	−0.011	13	0.27	0.7914	0.9576
	CPD+VS	Class Completeness	0.14	30	**2.933**	**0.0064**	**0.0066**
		CD Completeness	0.106	26	**2.392**	**0.0242**	0.0582
		Class Correctness	0.08	25	**2.73**	**0.0114**	**0.021**
		Redundancy	−0.068	26	−1.904	0.0682	0.2122
2.3	CPD	Class Completeness	0.193	12	**3.188**	**0.008**	**0.0382**
		CD Completeness	0.136	10	2.203	0.0514	0.0814
		Class Correctness	0.054	9	1.476	0.0175	0.22
		Redundancy	−0.111	12	**−2.423**	**0.0326**	**0.0446**
	VS	Class Completeness	0.056	15	0.768	0.454	0.532
		CD Completeness	0.017	16	0.277	0.7854	0.7904
		Class Correctness	0.045	13	1.765	0.1002	0.0934
		Redundancy	−0.053	16	−1.059	0.3056	0.2866
	CPD+VS	Class Completeness	0.124	29	**2.326**	**0.0274**	**0.0344**
		CD Completeness	0.076	30	1.634	0.1128	0.1214
		Class Correctness	0.054	30	**2.112**	**0.0432**	0.0892
		Redundancy	−0.08	30	**−2.323**	**0.0272**	**0.0338**

calculation of *CD Correctness*, is then undefined due to a division by zero. Therefore, we exclude *Association Correctness* and *CD Correctness* from our analysis.

4.2.1.1. Univariate analysis. The participants applying UCM_R consistently performed better than the participants applying UCM_UR in terms of all the four measures in both Experiment 1 and Experiment 2. By comparing the global means for the two experiments, we observed that the results of Experiment 2 are overall better than those of Experiment 1 regarding *Class Completeness*, *CD Completeness*, and *Class Correctness*. However it is the opposite for *Redundancy*; Experiment 2 yielded higher mean values than Experiment 1. As discussed in Section 3.4, participants in Experiment 2 were given twice as much time to derive class diagrams and we were expecting to obtain more complete class diagrams and, as a result, more redundant classes.

Table XIX reports on the results of a two-tailed t-test to assess the statistical significance of differences of the four CD quality measures between UCM_R and UCM_UR. Equivalent non-parametric tests results (Wilcoxon rank sum test [Wohlin and Wesslen et al. 2000]) are also reported to confirm the validity of t-test results. Each row reports on each *CD* quality measure for each system (CPD or VS) or the two systems together (CPD+VS) for Experiment 1 (due to the small number of observations). Columns show the mean differences, degree of freedom (DF), the calculated t-value, and the corre-

Table XX. Summary of ANOVA Tests – CD

Main effects	Experiment	Class Completeness	CD Completeness	Class Correctness	Redundancy
System (in favor of VS)	2.1	No	No	Yes	Yes
	2.3	Yes	Yes	Yes	No
Order (in favor of Lab 2.3)		No	No	Yes	Yes

sponding p-value of the two-tailed *t*-test and corresponding probability for the non-parametric test.

In Experiment 1, though the differences are not significant, the participants using UCM_R performed slightly better in terms of *Class Completeness*, *CD Completeness*, and *Redundancy* than those using UCM_UR. However, there is a statistically significant difference regarding *Class Correctness*: UCM_R yielded significantly higher quality class diagrams than UCM_UR. In lab 2.1, Experiment 2, UCM_R are better in terms of all four measures than UCM_UR. There are statistically significant differences regarding *Class Completeness* and *Class Correctness*. There is, however, a conflict between the results of the *t*-test and the non-parametric Wilcoxon test regarding *CD Completeness*. In lab 2.3, Experiment 2, again, UCM_R outperformed UCM_UR in terms of all the four measures. Statistically significant differences are observed in terms of *Class Completeness*, *Class Correctness*, and *Redundancy*, which is not consistent with the results of lab 2.1. This inconsistency is due perhaps to interaction effects between *Order* (2.1 vs. 2.3) and *Method* (UCM_R vs. UCM_UR). A two-way ANOVA test (Section 4.2.1.2) was performed to examine the impact of lab order on the dependent variable *CD*.

Various statistically significant differences are observed for CPD-only *t*-tests and no statistically significant differences are observed for VS-only *t*-tests, which indicates that it is necessary to perform a two-way ANOVA test to assess interaction effects between *System* (CPD vs. VS) and *Method* (UCM_R vs. UCM_UR). Besides, in Experiment 2, the mean values of the VS system are consistently higher (*Class Completeness*, *CD Completeness* and *Class Correctness*) or lower (*Redundancy*) than those of the CPD system. Therefore we conducted a two-way ANOVA analysis (Section 4.2.1.2) to assess the impact of *System* (CPD vs. VS) on *CD*.

4.2.1.2. Interaction effects. A Two-way ANOVA was performed to test both the interaction between *Method* and *System* and between *Method* and *Order* for Experiment 2. No significant interaction effects were identified for any of the four measures. However, significant main effects of *System* and *Order* were found on various measures. Results are summarized in Table XX.

For lab 2.1, the results show significant main effects of *System* on *Class Correctness* and *Redundancy* (in favor of VS) and no significant interaction effect on any of the four measures. Significant main effects of *System* are observed in lab 2.3 for *Class Completeness*, *CD Completeness*, and *Class Correctness* (in favor of VS). No significant interaction effect is identified in lab 2.3 for any of the four measures. Least-square means plots used to visualize the main and interaction effects of *Method* and *System* in lab 2.1 and lab 2.3 are provided in Yue et al. [2010b] for reference. Participants performed better on VS than on CPD perhaps because the UCMod of VS contains five use cases while the UCMod of CPD has six use cases, the latter thus requiring more time to understand.

Lab order was the second factor for which we studied the interaction effect on *CD* to account for learning effects. We assume that participants would tend to perform better on lab 2.3 than lab 2.1, regardless of other factors. We performed a two-way ANOVA analysis to assess the impact of lab order on the four measures of CD. Statistically significant differences are found for the main effects of *Order* in terms of *Class Correctness* and *Redundancy* (in favor of Lab 2.3), but no significant effect was observed for the interaction of *Method* and *Order*. We also observed that the participants performed better in Lab 2.3 than in Lab 2.1 in terms of all the four measures.

4.2.2. Quality of Analysis Sequence Diagram (SD). In this section, we discuss the impact of *Method* (with two treatments *UCM_R* vs. *UCM-UR*) on the dependent variable *SD* in Section 4.2.2.1, which is determined by different measures (Section 3.6.4). Next, in Section 4.2.2.2, we investigate the possible interactions between *Method* and possible interaction factors: *System* (*CPD* vs. *VS*) and *Order* (2.2 vs. 2.4). A detailed description of the analysis is provided in Yue et al. [2010b].

In the rest of the analysis, we report on a selected set of the *Completeness*, *Correctness*, *Redundancy*, and *BCE Consistency* measures discussed in Section 3.6.4: *Message Completeness*, *SD Completeness*, *Message Correctness*, *SD Correctness*, and *BCE Consistency*. The rationale is that *Message* is one of the most important elements of sequence diagrams and *SD Completeness* and *SD Correctness* indicate the overall completeness and correctness of a sequence diagram, including the completeness and correctness of its interaction uses and combined fragments. Therefore it is not necessary to report on the completeness and correctness of interaction uses and combined fragments separately. We also exclude *Redundancy* from the analysis of sequence diagrams as we observed very few sequence diagrams that contained redundant messages.

4.2.2.1. Univariate analysis. We performed univariate analysis to assess the isolated effect of *Method* on the five selected quality measures for *SD*. We observed that UCM_R is better than UCM_UR in terms of most of the measures and for both systems in both lab 2.2 and lab 2.4.

Table XXI reports on the results of a two-tailed *t*-test for each system to assess the statistical significance of the difference for the five selected quality measures between UCM_R and UCM_UR. Non-parametric tests (Wilcoxon rank sum test [Wohlin and Wesslen 2000]) were also performed to double-check the validity of *t*-test results. In lab 2.2, there is no statistically significant difference observed. In lab 2.4, statistically significant differences regarding *SD Completeness* and *Message Correctness* are found based on the CPD+VS data points.

4.2.2.2. Interaction effects. We performed a two-way Analysis of Variance (ANOVA) to assess possible interactions between *Method* and *Order* and between *Method* and *System*. No significant interaction effects were identified for any of the four measures. However, significant main effects of *System* and *Order* were found on various measures. The results are summarized in Table XXII. As shown in the table, only significant main effect of *System* on measure *SD Completeness* was observed in lab 2.2 in favor of CPD. To account for learning effects, the interaction effect of lab order on *SD* was also studied. We assume that participants would tend to perform better on lab 2.4 than lab 2.2, regardless of other factors. The results show a statistically significant impact of lab order through interactions with *Method* on *Message Completeness*, *SD Completeness*, *Message Correctness*, and *SD Correctness*, but not on *BCE Consistency*. The most plausible explanation is that participants improved their UML sequence diagram modeling skills from experience in lab 2.2 and therefore performed better in lab 2.4. This tentative explanation is confirmed by the participants when filling pre-lab questionnaires [Yue et al. 2010b], reporting on how well they felt they mastered UML sequence diagram modeling.

4.2.3. Correctness of Responses to the Comprehension Questionnaire. The dependent variable *QC* denotes the correctness of responses from a comprehension questionnaire (Sections 3.6.3 and 3.6.4). In this section, we report on two-tailed *t*-test results using the

Table XXI. Two-Tailed t-test and Wilcoxon Test – SD

		t-test					Wilcoxon test
Lab	System	Measures	Mean difference (UCM_R – UCM_UR)	DF	t-value	p-value	Prob > |Z|
2.2	CPD	Message Completeness	0.041	8	0.8061	0.4438	0.201
		SD Completeness	0.056	11	0.6955	0.5012	0.5624
		Message Correctness	0.237	13	1.9451	0.0738	0.2948
		SD Correctness	0.156	11	1.872	0.0878	0.0636
		BCE Consistency	0.046	8	0.7322	0.4856	0.9
	VS	Message Completeness	0.022	10	0.3419	0.7392	0.7746
		SD Completeness	−0.009	9	−0.1193	0.9076	0.5672
		Message Correctness	0.001	10	0.0085	0.9834	0.9432
		SD Correctness	0.045	10	0.6804	0.5124	0.6162
		BCE Consistency	0.044	5	1.45	0.2056	0.158
	CPD+VS	Message Completeness	0.032	22	0.792	0.437	0.2584
		SD Completeness	0.025	26	0.4094	0.6856	0.7952
		Message Correctness	0.128	25	1.4766	0.1524	0.1814
		SD Correctness	0.107	22	1.9275	0.0672	0.076
		BCE Consistency	0.045	16	1.247	0.2306	0.3742
2.4	CPD	Message Completeness	0.063	8	0.8596	0.416	0.5736
		SD Completeness	0.155	12	1.4738	0.1666	0.1752
		Message Correctness	0.128	11	1.0345	0.324	0.3006
		SD Correctness	0.016	11	0.1726	0.8662	1
		BCE Consistency	0.008	12	0.0751	0.9514	1
	VS	Message Completeness	0.124	10	1.5246	0.1572	0.3678
		SD Completeness	0.154	12	**2.5168**	**0.0268**	**0.0204**
		Message Correctness	0.229	8	2.0051	0.0778	0.1146
		SD Correctness	0.066	12	1.3277	0.2098	0.3436
		BCE Consistency	0.008	14	0.495	0.6286	0.7012
	CPD+VS	Message Completeness	0.098	19	1.798	0.088	0.1546
		SD Completeness	0.156	28	**2.7023**	**0.0116**	**0.0076**
		Message Correctness	0.186	26	**2.3462**	**0.0268**	**0.0304**
		SD Correctness	0.05	28	0.97	0.3404	0.3078
		BCE Consistency	0.014	26	0.259	0.7978	0.7433

Table XXII. Summary of ANOVA Tests – SD

Main effects	Exp	Message Completeness	SD Completeness	Message Correctness	SD Correctness	BCE Consistency
System (in favor of CPD)	2.2	No	Yes	No	No	No
	2.4	No	No	No	No	No
Order (in favor of lab 2.4)		Yes	Yes	Yes	Yes	No

Table XXIII. Descriptive Statistics of QC – Experiment 1, Experiment 2: lab 2.2, and lab 2.3

	Experiment 1		Lab 2.2		Lab 2.3	
Methods	Mean	Size	Mean	Size	Mean	Size
UCM_R	0.913	12	0.816	18	0.852	16
UCM_UR	0.527	8	0.545	15	0.494	18
All Methods	0.72	20	0.693	33	0.662	34

Table XXIV. t-test – QC – Experiment 1, Experiment 2: lab 2.2, and lab 2.3

Experiment	Mean difference (UCM_R – UCM_UR)	DF	t-value	p-value
1	0.387	8	**5.189**	<**0.0008**
2.2	0.271	29	**4.4**	<**0.0002**
2.3	0.356	26	**8.4**	<**0.0002**

factor *Method*. The descriptive statistics of the measure are presented in Table XXIII and the t-test results are given in Table XXIV. The t-test result shows a significant difference between the two treatments in the expected direction for all the experiments, thus indicating an increased understanding due to restriction rules and the template. Nonparametric test results are not very different from the t-test results and are therefore not presented.

4.3. Summary of Analysis Results

As discussed in the previous sections, we have evaluated RUCM from various aspects. With respect to the usage of RUCM, we evaluated each restriction rule in terms of its understandability, applicability, restrictiveness, and error rate.

—*Error rate:* we observed that the two most complex restriction rules have the highest error rates of 21% and 25%, and 14 out of 26 restriction rules have an error rate less than 5%. Additionally, we also observed that most of the restrictions on the use of natural language have an error rate lower than the restrictions on the use of control structures.

—*Understandability, Applicability and Restrictiveness:* most of the restriction rules are observed to have reasonably high understandability (more than 80%). More than 80% of the participants agree that most of the restriction rules are easy to apply and more than half of the rules are not restrictive.

These results are very encouraging as they suggest the restriction rules are in general easy to understand and apply, and that they are not perceived to be too restrictive. The restrictions rules that are the most difficult to apply constrain the specification of control flow and we strongly believe appropriate training (e.g., focusing on error-prone rules and the ones with lower scores on understandability, applicability and restrictiveness) and tool support (e.g., enforcing the usage of control structure keywords) would be easy to put in place and would tremendously reduce error rates and improve the overall applicability of RUCM. We also believe that informing the participants of the motivation and rationale of the restriction rules could possibly further improve their overall applicability.

In terms of the impact of use case modeling approaches, either RUCM (UCM_R) or a more traditional use case specification procedure (UCM_UR), on the quality of UML class and sequence diagrams, we mainly have the following observations.

—*Quality of analysis class diagram (CD):* the participants applying UCM_R consistently performed better than the ones applying UCM_UR in both experiments. In Experiment 1, significant differences were identified for *Class Correctness* in favor of UCM_R. In Experiment 2, UCM_R leads to significantly better diagrams than UCM_UR in terms of *Class Completeness* and *Class Correctness*. No significant interaction effects between *Method* and *System* and between *Method* and *Order* were identified for any dependent variables, though significant main effects of *System* and *Order* were found on various measures.

—*Quality of analysis sequence diagram (SD):* the participants applying UCM-R performed better than the ones applying UCM_UR in terms of most of the measures and for both systems. Statistically significant differences regarding *SD Completeness* and *Message Correctness* are found based on the combined observations of the two systems. Similarly to CD, no significant interaction effects between *Method* and *System* and between *Method* and *Order* were identified for any dependent variables, but significant main effects of *System* and *Order* were found on various measures.

—*Correctness of response to the comprehension questionnaire (QC):* significant differences between the two treatments in favor of UCM_R for all the experiments were identified.

Once again these results are very encouraging since they indicate that RUCM has an overall greater positive impact on the quality of manually derived analysis models than a more traditional use case modeling procedure.

5. DISCUSSION

In this section, we first discuss the threats to validity of the experiments (Section 5.1). Then we discuss the practical implications and limitations of this research (Section 5.2).

5.1. Threats to Validity

Two main threats to external validity are related to our experiment, and are typical to 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, in terms of the size of the systems we used?

Regarding issue 1), recall that in Task 1 the participants designed UCMods by applying the restriction rules and the use case template we proposed. 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, either participants or professional requirements engineers will likely require training. The participants of our experiment are 4th year software and computer engineering students who have received extensive training in use case modeling in previous courses. In addition, they were given a one-and-half-hour lecture and an assignment specifically focusing on how to apply the restriction rules and the use case template. In our context, the main difference between students and professional requirements engineers, is that the latter could have more experience on designing UCMods, they could be more familiar with the domain, and thus we assume that they would probably apply more effectively the restriction rules and the use case template than students given the same amount of training and time to perform the task. Thus, we believe that 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' understandability, applicability, and restrictiveness. As for Task 2, the students derived analysis models from the UCMods, with or without restrictions and template. This task is usually performed by system analysts. Again, our 4th year software and computer engineering students had received extensive training on software modeling with UML, through several courses, and though they did not derive perfect class diagrams (specifically, poor use of class associations, though class correctness is good), results were comparable to what some have observed in many software development environments [Lange et al. 2006]. In addition, some studies [Arisholm and Sjoberg 2004; Holt et al. 1987; Höst et al. 2000] suggest students may be representative of some professionals. The performance of trained software engineering students was compared to the one of professional developers. Differences in performance were not statistically significant when compared to junior and intermediate developers, thus leading to the conclusion that there is no evidence that students trained for a specific task may not be used as subjects in place of professionals.

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 comprehension questionnaire for Task 1 (Section 3.5.1.3) was not involved in any comparison so it does not bear on construct validity. The questions of the comprehension questionnaire for Task 2 (Section 3.5.2.3) were designed to be answerable from the UCMods with or without restrictions, therefore introducing no bias for any of the treatments. The same quality measures are used to measure the students' class and sequence diagrams derived from the UCMods with or without restrictions and therefore no bias is introduced as well when evaluating these diagrams.

Three students presented problems related to internal validity (Section 4) and these three data points were therefore excluded from the analysis. The participants belonging to different groups were monitored to ensure they would not access each other's documents during the entire lab duration. By doing so, we also limited the threat on the internal validity of the experiments.

5.2. Practical Implications and Limitations

From the experiment results reported in Section 4.1, we observed that RUCM restriction rules are overall easy to understand, apply and not too restrictive to be applicable in practice. We also observed that most of the restriction (rules) on the use of natural language have lower error rates than the ones on the use of control structures. Therefore we believe that with appropriate tool support, the applicability of RUCM restriction rules can be further improved since adequate tool support can be used to enforce the proper usage of keywords specified in the restrictions on the use of control structures.

It is worth noting that inconsistent use case specifications in a use case model may lead to low quality class and sequence diagrams. However, we believe that inconsistencies among use case specifications can be reduced if the RUCM restrictions are properly applied, and that some inconsistencies can be partly and automatically detected by checking a use case model against the RUCM restriction rules. In sum, with proper tool support, RUCM can be better applied in practice. Developing such a tool is part of our future work.

We made an effort to devise a set of objective metrics to measure the quality of class and sequence diagrams by comparing them with reference models (Sections 3.6.3 and 3.6.4), that are domain independent (e.g., we used case study systems from different domains), organization independent (e.g., we used textbook and industry case study systems), software engineer independent (e.g., we used students and experts), while reducing subjective perceptions to the maximum extent possible. With such a set of objective metrics, identical results should be obtained by different persons measuring models. In addition, these metrics are general: they are not specific to our experiments and one should be able to apply them in other contexts.

Table XXV. Summary of Use Case Templates

Fields	Cockburn [2001]	Jacobson et al. [1992]	Kruchten [2003]	Kulak et al. [2000]	Larman [2004]	Liu [2003]	Somé [2006]	Insfrán et al. [2002]	RUCM
Use case name	*	*	*	*	*	*	*	*	*
Description	*	*	*	*	*	*	*	*	*
Precondition	*	*	*	*	*	*	*		*
Postcondition	*	*	*	*	*	*	*		*
Basic flow	*	*	*	*	*	*	*	*	*
Alternative flow	*	*	*	*	*	*	*	*	*
Primary actor	*						*	*	*
Secondary actor							*	*	*
Scope	*								
Level of abstraction	*								
Stakeholders and interests	*								
Special requirements			*					*	
Extension points			*						*
Exception path[1]				*					
Cross-reference to high-level requirements								*	
Extension	*								
Variation	*								

1. Exception paths are distinguished from alternative flows since they are paths taken when errors occur.

6. RELATED WORK

Several streams of research relate to our work: use case templates (Section 6.1), restriction rules (Section 6.2), and experimental evaluations of use case modeling approaches (Section 6.3).

6.1. Use Case Template

The concept of use case has been first introduced by Ivar Jacobson in 1986 to capture functional requirements [Jacobson 1987]. Since then, use cases have been widely accepted and use case modeling techniques have evolved and matured. The concept of use case is part of the UML (though UML does not support use case specification), which provides a use case diagram to specify relations between use cases and between use cases and actors. Use case modeling, among other requirements specification approaches, is more than a requirements specification technique; it drives the whole software development process including activities such as analysis, design, and test. If a software development process starts from use cases, then it is referred to as use case-driven software development [Jacobson et al. 1992].

It is a common practice to follow a use case template to structure UCSs, thereby helping their writing, reading and reviewing. Various use case templates (e.g., Cockburn [2001], Jacobson et al. [1992], Kruchten [2003], Kulak et al. [2000], and Larman [2004]) have been suggested in the literature to satisfy different application contexts and purposes. They share common fields such as *use case name*, *description*, *basic flow*, and *alternative flows*, as shown in Table XXV, where all the fields of the use case templates of the related work are summarized. In the table, "*" indicates that a specific template (column) contains a specific field (row).

In addition to capturing requirements, use cases can also facilitate the manual or automated derivation of an initial analysis model. The systematic literature review [Yue et al. 2011b] we conducted to examine techniques that transform textual requirements into analysis models revealed that six approaches require use cases. As shown in Table XXV, the approach proposed in Liu [2003] relies on the RUP use case template [Kruchten 2003]. The use case template proposed in [Somé 2006] contains eight fields. An enriched use case template is proposed in Insfrán et al. [2002], and is composed of three sections: use case summary, basic flows, and alternative flows. The use case summary section is further divided into four subsections: use case name, actors, cross-reference, and an overview of the use case purpose (may be used to describe nonfunctional requirements). The cross-reference section is used to link the use case to high-level requirements. A three-column structure (i.e., general, actor/system communication, and system response) is introduced in this use case template to structure the steps of flow of events. The general column describes general activities that are not supposed to be implemented by the system but can help users better understand the use case. Steps in the actor/system communication column specify actions performed by actors when interacting with the system. The system response column represents reactions of the system, including changes of state.

The use case template we propose in this article (Section 2.1) contains fields similar to those encountered in conventional use case templates but with a few variations on the structure of the flow of events. The motivation is to support the automated generation of analysis models and to further reduce ambiguity. Our objective is to propose a new use case template that not only complies as much as possible with conventional use case templates but also facilitates the process of deriving analysis models. Therefore, 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. (They could easily be added though.) (3) We excluded the fields that, on the one hand may increase the precision of UCSs but, on the other hand require that the designer provides much more information than what is actually needed 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. For example, the semantics of the three-column steps modeling style proposed in Insfrán et al. [2002] (discussed previously) can actually be automatically derived from UCSs by grammatically analyzing each sentence; therefore we made a design decision not to include this style into our use case template. (4) Six interactions types (five from Cockburn [2001], one we newly propose in this work) are suggested to describe action steps of flow 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: the postcondition of the use case is simply the disjunction of all those postconditions. By doing so, our postconditions are simpler to understand.

6.2. Restriction Rules

In Yue et al. [2011b], we summarize and classify the restriction rules (also called writing guidelines) applied in Śmiałek et al. [2007], Somé [2006], Subramaniam et al. [2004], and Wahono and Far [2002], which propose transformations from requirements to analysis models. In this article, we propose a total of 26 restriction rules on the use of

Natural Language (NL) to document a UCS that complies with our use case template. Some of these restrictions are the results of the systematic review [Yue et al. 2011b] we conducted (see further details in Section 2.2); others are heuristics suggested in the literature on writing use cases (e.g., Achour et al. [1999], Bittner and Spence [2002], and Cockburn [2001]); last some of them are derived from our experience in writing UCSs when attempting to reduce ambiguity and facilitating automated transition to analysis models. None of the related work we looked at relies on a set of rules as complete as the one we suggest.

Existing guidelines are recommended, based on practitioners' experience in writing UCSs, to reduce ambiguities of UCSs or to facilitate the process of deriving analysis models from them. We reused some of them, excluded others, recommended new ones, and classified all the rules. For example, we excluded the rules that constrain grammatical sentence structures, such as only allowing certain types of structures such as subject-verb-object (e.g., Achour et al. [1999] and Cox [2002]), because these structures can be automatically obtained by grammatically analyzing each sentence using a NL parser. We also excluded the rules that put excessive constraints on wording. For example, one such rule suggests using "is a kind of," "is specification of," or "is generalization of" to describe inheritance between the subject and object of a sentence.

Additionally, we explicitly describe why each of our restriction rules is needed either to reduce ambiguities or facilitate the process of deriving analysis models, a crucial piece of information that is often omitted in the literature. We also indicate how and where to apply (Section 2.2) 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 NL parsers (e.g., The Stanford Parser version 1.6[3]) 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 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 2.2).

6.3. Empirical Evaluation

Some empirical studies have been conducted to evaluate the impact of applying restriction rules on the quality of UCSs. Achour et al. [1999] investigated the effectiveness of the CREWS rules in terms of the completeness and structuredness of UCSs: UCSs were evaluated by comparing them to "expert" UCSs developed by the authors of the paper. The experiment results show that the application of the rules produced more complete and better structured UCSs. Phalp et al. [2007b] conducted an empirical study to compare two sets of writing rules: the CREWS rules and CR rules [Cox 2002] (leaner than the CREWS rules). The overall quality of UCSs was evaluated based on seven quality factors, referred as the "7Cs of communicability" [Phalp et al. 2007a]; coverage (a use case should contain all the required information), cogent (a sentence should follow a logical path), coherent (sentences should be all connected by, for example, repeating a noun), consistent abstraction, consistent structure, consistent grammar, and consideration of alternatives. The experiment results from Phalp et al. [2007b] show that the leaner set of rules (the CR rules) results in less learning overhead than the CREWS rules and performs at least as well as the CREWS rules, in terms of the overall quality of produced UCSs. Anda et al. [2001] conducted a similar experiment to compare three different sets of guidelines: minimal guidelines (guidelines on identifying actors and use cases), template guidelines (a commonly applied actor and use case template based

[3]http://nlp.stanford.edu/software/lex-parser.shtml.

on templates proposed in Cockburn [2001], Kulak and Guiney [2000], and Schneider and Winters [1998]), style guidelines (modified version of the CREWS rules, focusing on the documentation of the flow of events of each use case). UCSs were evaluated in terms of their understandability, usefulness, and quality. The experiment results show that the template guidelines led to the highest understandability, usefulness, and overall quality, and that the style guidelines performed better than the minimal guidelines. The authors also suggest that combining the style guidelines with the template guidelines might further improve quality attributes of UCSs to compare with independently applying the template or style guidelines. This is exactly the case of our RUCM.

All these experiments evaluated restriction rules as a whole; none of them evaluated each rule individually. We however evaluate each of our restriction rules both individually and as a whole. By doing so, we are able to tell which rule(s) are difficult to apply and therefore where extra focus and significant practical exercises during training should be given. It is also worth noticing that all of these related works assess the quality of UCSs against some quality criteria (e.g., understandability, structuredness, completeness), rather than test the ability of subjects to extract information from UCSs such as deriving analysis models from UCSs. In this article, we report on two controlled experiments, which evaluate the impact of our restriction rules and use case template both on the understandability of UCSs and the quality of analysis models manually generated from them in terms of correctness, completeness, and redundancy.

An empirical study on the role of use cases in the construction of class diagrams is reported in Anda and Sjoberg [2005]: two controlled experiments were performed, with students and professionals as subjects, respectively. Two use case driven approaches were compared: one is to derive classes by analyzing the use cases (derivation technique); while the other is to identify classes from a textual requirements specification and subsequently apply the use case model to validate the resulting class diagram (validation technique). Results show that the derivation technique resulted in class diagrams with a significantly better structure than the validation technique in the student experiment and slightly better structure in the experiment with the professionals. They also show that results depend on the categories of the developer applying a technique (either students or professionals) and on the tool with which the technique is applied (pen and paper, or modeling tools). The class diagrams derived by the subjects were evaluated with three dependent variables: *Completeness* measured in terms of how much of the functionality described in requirements was actually implemented in a class diagram, *Structure* measured in terms of cohesion and coupling, and *Time* spent on obtaining class diagrams. This measurement is different from ours. We evaluate the quality of the class diagrams derived by the subjects from three aspects: *Completeness*, *Correctness*, and *Redundancy*, by comparing them with reference class diagrams (Section 3.6.3) which are considered to be mostly correct and complete.

A subjective and experience-based comparison of three approaches that guide users to manually derive class diagrams from use cases is presented in Liang [2003]. These three approaches are three processes that, respectively, require manually analyzing (1) use case goals defined by human actors of use cases (goal-oriented), (2) nouns contained in use case specifications (noun-oriented), and (3) scenarios described by flows of events in use case specifications (scenario-oriented). In our controlled experiments, the subjects were asked to directly derive class diagrams from use case models including both use case diagrams and use case specifications and no specific process was prescribed during the class diagram derivation process as our objective was to assess RUCM as opposed to a process. In addition, Liang's work does not evaluate the quality of the resulting class diagrams as we do, but subjectively and qualitatively assesses the effectiveness of the three processes and the stability of the derived class diagrams in the presence of changes to use case goals, specifications, and scenarios.

The evaluation results show that the goal-oriented approach is more effective than the other two in the sense that it does not need to identify candidate classes (as opposed to the noun-oriented approach), and it is independent from any specific use case scenarios (in contrast to the scenario-oriented approach). It however induces additional cost for the design of “use case-entity diagrams” as intermediate models. The goal-oriented approach is considered to be more stable than the other two because the goals of a use case considered to be unique, whereas the authors state that there might exist many use case specifications for a use case and many scenarios contained in a use case such that the noun-oriented and scenario-oriented approaches end up not being as stable as the goal-oriented approach.

A few approaches like TUCCA [Belgamo et al. 2005] and perspective-based reading [Maldonado et al. 2006] could be seen as requirements analysis techniques related to RUCM; however they are more properly described as requirements inspection techniques, as their analysis is focused solely on defect detection in a requirements specification. By comparison, RUCM aims to aid the transition to an analysis model. An empirical evaluation of TUCCA is presented in Belgamo et al. [2005], comparing TUCCA with other two requirements inspection techniques: checklist and perspective-based reading, which give the reviewer procedures to follow during a requirements inspection process. As one can see, this evaluation has different objectives than ours.

There also exists some works about measuring the quality of models. Xu et al. [2004] and Genero et al. [2000] proposed a structural complexity measure for UML class diagrams to evaluate their quality. Reißing [2001] proposed a model, based on the UML metamodel, for OO design measurement. This model defines structural metrics of a class diagram for the purpose of automating evaluation of UML class diagrams. Marchesi [1998] proposed a set of metrics for UML use case and class diagrams. The objective is to allow early estimate of development efforts and implementation time, etc. Kim and Boldyreff [2002] proposed a set of metrics for UML class and use case diagrams. They also proposed metrics for measuring messages in UML sequence diagrams. A set of metrics are proposed in Cruz-Lemus et al. [2010] to measure the complexity of UML state machine diagrams. In Major et al. [1999] and Reinhartz-Berger and Sturm [2008], the correctness of models is measured by their accuracy at representing the information specified in requirements while completeness is measured by assessing the extent to which models capture sufficient information for the established goals. None of these approaches actually measures the quality of UML class and sequence diagrams by comparing them with reference models as we do. By doing so, our measurement is more objective since a set of precise metrics can be defined by comparing the experiment diagrams with a “correct” solution.

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, the UCSs are expected to be the least ambiguous possible, to support either the manual or automated generation of analysis models. In this article, we propose a use case modeling approach, referred to as RUCM, which is composed of 26 well-defined restriction rules and a use case template. Its purpose is to restrict the use of natural language when users document UCSs in order to reduce ambiguity and also to facilitate the (automated) transition towards analysis models.

Two controlled experiments have been conducted, in the context of a fourth-year Software Engineering course, to evaluate whether RUCM is easy to apply while developing use case models and whether it helps human analysts to manually derive higher quality analysis models. Each restriction rule is evaluated in terms of its understandability, applicability, restrictiveness, and error rate. The experiment results indicate that our 26 restriction rules are easy to apply and with focused training on the rules receiving higher error rates and appropriate tool support (e.g., enforcing the usage of keywords and the definition of compulsory alternative flows), error rates can be expected to further decrease. Based on these results, we are therefore confident that trained engineers are capable of properly applying our restriction rules and template and obtain UCSs from which to derive analysis models.

Another goal of the controlled experiments was to evaluate whether RUCM 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 and sequence diagrams is evaluated from the viewpoints of their correctness, completeness, and redundancy. The results show that RUCM yields significant improvements across all experiments regarding the correctness of derived class diagrams, though no significant difference was observed in terms of their completeness and redundancy. The results also show that RUCM leads to significant improvement on the diagram completeness and message correctness of derived sequence diagrams during the second lab of the second experiment. No significant improvements were observed in the first lab, mostly because participants improved their UML sequence diagram modeling skills by gaining experience in the first lab, and therefore performed better in the second lab. Furthermore, our approach resulted in a large improvement in term of comprehension of the use case models as measured by a carefully designed questionnaire.

RUCM is just a restriction in the usage of regular use cases, and therefore cannot be worse than them in terms of capturing requirements, unless they are too restrictive for the analyst to appropriately express what she wants. The experiment results show, however, that they are not too restrictive and that, in addition, RUCM models have the precision required to manually derive high quality analysis models. So in short, RUCM is at least as effective at capturing requirements as regular use cases are.

This article reports on the first controlled experiments that evaluate the applicability, both individually and as a whole, of restriction rules used to document UCSs and that also evaluate the impact of these rules and associated template on the quality of constructed analysis class and sequence 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.

REFERENCES

Achour, C. B., Rolland, C., Maiden, N. A. M., and Souveyet, C. 1999. Guiding use case authoring: Results of an empirical study. In *Proceedings of the International Symposium on Requirements Engineering*. 36–43.

Anda, B., Sjoberg, D., and Jorgensen, M. 2001. Quality and understandability of use case models. In *Proceedings of the European Conference on Object-Oriented Programming.* Springer, 402–428.

Anda, B. and Sjoberg, D. I. K. 2005. Investigating the role of use cases in the construction of class diagrams. *Empirical Soft. Engin. 10*, 285–309.

Arisholm, E. and Sjoberg, D. I. K. 2004. Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. *IEEE Trans. Soft. Engin. 30*, 521–534.

Basili, V. R., Caldiera, G., and Rombach, H. D. 1994. The goal question metric approach. *Encyclopedia of Software Engineering 1*, 528–532.

Belgamo, A., Fabbri, S., and Maldonado, J. C. 2005. TUCCA: Improving the effectiveness of use case construction and requirement analysis. In *Proceedings of the IEEE International Symposium on Empirical Software Engineering*. 266–275.

Bittner, K. and Spence, I. 2002. *Use Case Modeling*. Addison-Wesley Boston.

Bruegge, B. and Dutoit, A. H. 2009. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall.

Cockburn, A. 2001. *Writing Effective Use Cases*. Addison-Wesley Boston.

Cox, K. 2002. Heuristics for use case descriptions, PhD Thesis, Bournemouth University, UK.

CRUZ-LEMUS, J., MAES, A., GENERO, M., POELS, G., AND PIATTINI, M. 2010. The impact of structural complexity on the understandability of UML statechart diagrams. *Inf. Sci. 180*, 2209–2220.

DOBING, B. AND PARSONS, J. 2006. How UML is used. *Commun. ACM 49*, 109–113.

GENERO, M., PIATTINI, M., AND CALERO, C. 2000. Early measures for UML class diagrams. *L'Objet 6*, 489–505.

GOMAA, H. 2000. *Designing Concurrent, Distributed, and Real-Time Applications with UML*. Addison-Wesley.

HOLT, R. W., BOEHM-DAVIS, D. A. AND SHULTZ, A. C. 1987. Mental representations of programs for student and professional programmers. In *Empirical Studies of Programmers II*. Ablex Publishing Corp., 33–46.

HÖST, M., REGNELL, B., AND WOHLIN, C. 2000. Using students as subjects - comparative study of students and professionals in lead-time impact assessment. *Empirical Soft. Engin. 5*, 201–214.

INSFRÁN, E., PASTOR, O., AND WIERINGA, R. 2002. Requirements engineering-based conceptual modelling. *Requirements Eng. 7*, 61–72.

JACOBSON, I. 1987. Object-oriented development in an industrial environment. In *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages & Applications*. ACM, 183–191.

JACOBSON, I. 2004. Use cases - yesterday, today, and tomorrow. *Soft. Syst. Model. 3*, 210–220.

JACOBSON, I., CHRISTERSON, M., JONSSON, P., AND OVERGAARD, G. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley.

KIM, H. AND BOLDYREFF, C. 2002. Developing software metrics applicable to UML models. In *Proceedings of the International Workshop on Quantitative Approaches in Object-Oriented Software Engineering*.

KRUCHTEN, P. 2003. *The Rational Unified Process: An Introduction*. Addison-Wesley.

KULAK, D. AND GUINEY, E. 2000. *Use Cases: Requirements in Context*. ACM Press.

LANGE, C. F. J., CHAUDRON, M. R. V., AND MUSKENS, J. 2006. In practice: UML software architecture and design description. *IEEE Softw. 23*, 40–46.

LARMAN, C. 2004. *Applying UML and Patterns*. Prentice-Hall.

LIANG, Y. 2003. From use cases to classes: a way of building object model with UML. *Inf. Softw. Tech. 45*, 83–93.

LIU, D. 2003. Automating transition from use cases to class model. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*. 125.

MAJOR, M. L. AND MCGREGOR, J. D. 1999. Using guided inspection to validate UML models. In *Proceedings of the 24th Annual IEEE/NASA Software Engineering Workshop*. 485–507.

MARCHESI, M. 1998. OOA metrics for the Unified Modeling Language. In *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*. 67–73.

MALDONADO, J. C., CARVER, J., SHULL, F., FABBRI, S., DORIA, E., MARTIMIANO, L., MENDONCA, M., AND BASILI, V. 2006. Perspective-based reading: A replicated experiment focused on individual reviewer effectiveness. *Empirical Soft. Engin. 11*, 119–142.

OMG 2005. UML 2.0 Superstructure Specification Object Management Group.

OPPENHEIM, A. N. 1992. *Questionnaire Design, Interviewing, and Attitude Measurement*. Pinter Pub Ltd.

PHALP, K. T., VINCENT, J., AND COX, K. 2007a. Assessing the quality of use case descriptions. Software Quality Journal 15, 69–97.

PHALP, K. T., VINCENT, J., AND COX, K. 2007b. Improving the quality of use case descriptions: empirical assessment of writing guidelines. *Softw. Qual. J. 15*, 383–399.

REINHARTZ-BERGER, I. AND STURM, A. 2008. Enhancing UML models: a domain analysis approach. *J. Datab. Manage. 19*, 74–94.

REISSING, R. 2001. Towards a model for object-oriented design measurement. In *Proceedings of the ECOOP Workshop on Quantative Approaches in Object-Oriented Software Engineering*. 71–84.

SCHNEIDER, G. AND WINTERS, J. P. 1998. *Applying Use Cases: A Practical Guide*. Addison-Wesley.

ŚMIAŁEK, M., BOJARSKI, J., NOWAKOWSKI, W., AMBROZIEWICZ, A., AND STRASZAK, T. 2007. Complementary use case scenario representations based on domain vocabularies. In *Proceedings of the International Conference on Model Driven Engineering Languages and Systems*.

SOMÉ, S. S. 2006. Supporting use case based requirements engineering. *Inf. Softw. Tech. 48*, 43–58.

SUBRAMANIAM, K., LIU, D., FAR, B. H., AND EBERLEIN, A. 2004. UCDA: Use case driven development assistant tool for class model generation. In *Proceedings of the International Conference on Software Engineering & Knowledge Engineering*. 324–329.

WAHONO, R. S. AND FAR, B. H. 2002. A framework for object identification and refinement process in object-oriented analysis and design. In *Proceedings of the 1st IEEE International Conference on Cognitive Informatics*. 351–360.

WOHLIN, C. AND WESSLEN, A. 2000. *Experimentation in Software Engineering: An Introduction*. Springer.

Xu, B., Kang, D., and Lu, J. 2004. A structural complexity measure for UML class diagrams. In *Proceedings of the International Conference on Computational Science*. 421–424.

Yue, T. 2010. Automatically deriving a uml analysis model from a use case model, PhD Thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, 350.

Yue, T., Ali, S., and Briand, L. 2011a. Automated transition from use cases to UML state machines to support state-based testing. In *Proceedings of the European Conference on Modeling Foundations and Applications*. 115–131.

Yue, T., Briand, L. C., and Labiche, Y. 2009. A use case modeling approach to facilitate the transition towards analysis models: Concepts and empirical evaluation. In *Proceedings of the International Conference on Model Driven EngineeringLanguages and Systems*. Springer, 484–498.

Yue, T., Briand, L. C., and Labiche, Y. 2010a. An automated approach to transform use cases into activity diagrams. In *Proceedings of the European Conference on Modeling Foundations and Applications*. Springer, 337–353.

Yue, T., Briand, L. C., and Labiche, Y. 2010b. Automatically deriving a UML analysis model from a use case model, Simula Research Laboratory, Tech. Rep. 2010–15.

Yue, T., Briand, L. C., and Labiche, Y. 2011b. A systematic review of transformation approaches between user requirements and analysis models. *Requirements Eng. 16*, 2, 75–99.

Received September 2010; revised June 2011, September 2011, October 2011; accepted October 2011