

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

*Tao Yue, Lionel Briand, and Yvan Labiche*



Software Quality Engineering Laboratory  
Department of Systems and Computer Engineering  
Carleton University, Ottawa, Canada

MODELS 2009, Denver, Colorado, USA  
October 8<sup>th</sup>, 2009

# Motivation

- **Use case modeling (UCM)** is commonly applied to structure and document requirements.
- **Use case specifications (UCSs)** are usually textual documents which inevitably contain ambiguities.
- **RUCM (Restricted Use Case Modeling)**
  - Ultimate goals:
    - G1: Make UCMs more comprehensible and precise.
    - G2: Automate the generation of analysis models from UCMs
- **Experimental evaluation of RUCM (G1)**
  - Is RUCM applicable and easier to understand?
  - Does it yield better models when used by humans?

# RUCM

- RUCM is composed of:
  - A **use case template** to structure UCSs.
  - A set of **restriction rules** to restrict the way that users write UCSs.
- RUCM is based in part on:
  - the results of a thorough literature review and
  - the need to devise automated transformation rules to analysis models

## RUCM – Use Case Template

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

## RUCM – Restriction Rules

- Classified all the rules into two categories:
  - Restriction on the use of Natural Language (NL)
  - Enforcing the use of specific keywords for specifying control structures

## RUCM – Restriction Rules (Cont.)

- R12
  - Use simple sentence only.
  - Reduce ambiguity and facilitate automated NL parsing.
  - Apply to all sentences in a UCS.
  - Example
    - *The system cancels the transaction and ejects the card.*
    - *The system cancels the transaction **MEANWHILE** the system ejects the card.*
- R21
  - Use the keyword **MEANWHILE** to model concurrency in sentences

# Objectives

- Applicability of restriction rules?
  - Understandability
  - Applicability
  - Restrictiveness
- Comprehensibility of RUCM models?
- Impact of RUCM on quality of derived analysis models?

# Subjects

- 34 students registered in a 4<sup>th</sup> Software Engineering course at Carleton University, Ottawa, Canada
- Courses:
  - Several OOAD courses
- Lectures on RUCM
- Assignment applying RUCM



# Applicability of rules

- Questionnaire, analysis of RUCM models
- Measurement
  - **Understandability, Applicability, and Restrictiveness**
  - **Error Rate**
- Results
  - **Error rates:** < 15% (most)
  - Over 80% students agree that most of the restriction rules are **understandable, easy to apply, not restrictive.**
  - Tool support and more training on “tough” rules?

## Comprehension, Quality– Hypotheses

<b>Dependent Variable</b>	<b>Null Hypothesis</b>	<b>Alternative Hypothesis</b>
Quality of analysis class diagram (CD)	$H_0: CD(UCM\_R) \leq CD(UCM\_UR)$	$H_a: CD(UCM\_R) > CD(UCM\_UR)$
Correct response rate of the comprehension questionnaire (QC)	$H_0: QC(UCM\_R) \leq QC(UCM\_UR)$	$H_a: QC(UCM\_R) > QC(UCM\_UR)$

- $H_0$ : there is no significant improvement in terms of two dependent variables when using RUCM models.
- $H_a$ : restricted use case models result in high quality analysis models or high correctness of responses to the comprehension questionnaire when compared to unrestricted use case models.

# Experiment Design

	Task	Group A		Group B	
		Group A1	Group A2	Group B1	Group B2
<b>Lab 1</b>	Defining UCSs	VS		CPD	
<b>Lab 2</b>	Deriving analysis models	CPD with restrictions	CPD without restrictions	VS with restrictions	VS without restrictions

- Two systems:
  - Car Parts Dealer (CPD)
  - Video Store (VS)

## DV Measurement

- Two dependent variables
  - CD: The quality of class diagrams
  - QC: The correctness of responses to the comprehension questionnaires

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

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

## Quality of class diagrams

- The *Completeness* of a class diagram is inversely related to the number of:
  - missing classes
  - missing associations
  - missing generalizations
- The *Correctness* of a class diagram is determined by:
  - the *Correctness* of matching classes
  - the *Correctness* of matching associations
- The *Redundancy* of a class diagram is computed as the ratio of redundant classes over all the classes of a student's class diagram.

## Quality of class diagrams

- The *Completeness* of a class is related to:
  - whether its stereotype is missed
  - whether there are missing attributes
  - whether there are missing operations
- The *Correctness* of a class is determined by:
  - whether it is correctly named?
  - whether it is correctly stereotyped?
  - whether it is correctly specified as abstract?
  - whether a single logical concept is represented?
  - whether a cohesive set of responsibilities is assigned to it

# Results and Analysis

Statistical *t*-test results

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

The students with treatment of UCM\_R performed:

- Statistically significant better regarding *Correctness* and *QC*.
- Slightly better in terms of *Completeness* and *Redundancy*, but not statistically significant, due perhaps to:
  - time constraints of the experiment
  - the small size of our sample

# Conclusion

- A new use case modeling approach (RUCM)
  - Goal: Automated analysis model generation
  - A use case template
  - A set of 26 well-defined restriction rules
- A controlled experiment was conducted
  - Our 26 NL restriction rules are easy to apply
  - RUCM leads to
    - Improved students' comprehension (of use case model)
    - Significant correctness improvements (class diagram)

# Current Status and Future work

- Tool: aToucan
  - Use RUCM models as inputs
  - Generate analysis class diagrams
  - Generate analysis sequence diagrams
  - Based on Kermeta and the Stanford NL parser
  - Technical Report:  
**“Automatically Deriving a UML Analysis Model from a Use Case Model”**, Technical Report SCE-09-09, Carleton University
- Future work:
  - Evaluate the impact of RUCM on the quality of analysis sequence diagrams
  - Replicate the experiment
    - Impact of giving more time to participants?

Thank you!

Any questions?

