RUCM

TAO YUE
tao@simula.no
Simula Research Laboratory

Restricted Use Case Modeling Approach

User Manual

April 2010

Preface

Use case modeling is commonly applied to document requirements. Restricted Use Case Modeling (RUCM) is a use case modeling approach, which is composed of a set of well-defined restriction rules and a new template. The goal of RUCM is to reduce ambiguity, improve understandability of use case models, and facilitate automated generation of analysis models. The approach has been experimentally evaluated to be applicable, and easier to understand. It yields better models when used by humans.

This document gives a description of RUCM, gives an example to illustrate how to apply RUCM, and also provides a quick reference in the end.

TABLE OF CONTENTS

Introduction

A use case model, including a use case diagram and a set of use case specifications, is commonly applied in practice to structure and document requirements.

Use case modeling, including use case diagrams and use case textual specifications, is commonly applied to structure and document requirements. Use Case Specifications (UCS) are usually textual documents complying with a use case template that, though helping read and review use cases, inevitably contains ambiguities. RUCM is a use case modeling approach, and its goal is to restrict the way users can document UCSs in order to reduce ambiguity, improve the understandability of use case models, and facilitate automated analysis to derive initial analysis models, which in the Unified Modeling Language (UML) [15] are minimally composed of class and interaction diagrams, and possibly other types of diagrams and constraints.

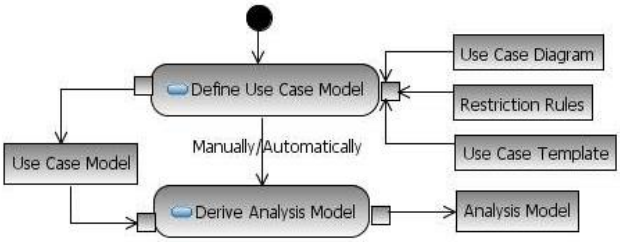

Figure 1 Defining use case model and deriving analysis model activities

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., [5]) to produce, to the extent possible, precise and unambiguous use case models, which is modeled as the first activity `Define Use Case Model` of the activity diagrams shown in Figure 1. Our use case modeling approach (RUCM) takes in input the use case diagram, the restriction rules, and the use case template. A use case diagram in UML [18], is used to represent relationships among actors and use cases; the restriction rules we defined are applied to restrict the way users can write UCSs; the use case template is the means to structure UCSs. With a use case model documented by applying RUCM, initial analysis models can then be derived as denoted by the activity `Derive Analysis Model`. This step is usually manually performed by system analysts but can also be automated, especially if inputted use case models are less ambiguous and thus automated analysis can be facilitated. In this document, we report on how to apply RUCM to define use case model (the first activity of Figure 1).

In this rest of the section, we first describe RUCM including the use case template and the restriction rules. Then we use a running example to illustrate how to define use case models by applying RUCM.

RUCM Use Case Template

A use case template is used to structure use case specifications. Our RUCM template is based in part on the results of a thorough literature review and the need to devise transformation rules to analysis models.

RUCM template integrates elements of existing related works and it contains fields that are commonly encountered in conventional templates, but seeks to better specify the structure of flows of events of a use case specification. RUCM template has eleven first-level fields (first column of Table 1). The last four fields are decomposed into second-level fields (second column of the last four rows).

Use Case Name

This field gives the name of the use case. It usually starts with a verb (e.g., *Withdraw Fund*) and should be consistent with the name of the same use case in the use case diagram.

Brief Description

This field summarizes the use case in a short paragraph. It captures the essence of the use case.

Precondition

The precondition of a use case specifies what must be always true before the use case begins.

Primary Actor

The primary actor of a use case is the principle actor which initiates the use case.

Secondary Actors

The secondary actors of a use case are the actors that the system relies on to accomplish the use case.

Dependency

This field specifies <<include>> and <<extend>> relationships of a use case to other use cases.

Generalization

This field specifies generalization relationships a use case to other use cases.

Basic Flow

The basic flow of a use case describes a main successful path that satisfies the interests of the stakeholders. It often does not include any condition or branching. It is recommended to describe separately the conditions and branching in alternative flows. A basic flow is composed of a sequence of steps and a postcondition. Each UCS can only have one basic flow.

The action steps can be one of the following five interactions:

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.

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 should be applied.

Alternative Flows

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). 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:

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 a 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 [3]) is an alternative flow that refers to any step in the reference flow.

Distinguishing different types of alternative flows makes interactions between the reference flow and its alternative flows much clearer. For specific and bounded alternative flows, a RFS (Reference Flow Step) section, specified as rule R19, is used to specify one or more (reference flow) step numbers. Whether and where the flow merges back to the reference flow or terminates the use case must be specified as the last step of the alternative flow. Similarly to the branching condition, merging and termination are specified by following restriction rules (R24 and R25—Section Restriction rules on the use of keywords for specifying control structures (R17-R25) and R26). By doing so, we can avoid potential ambiguity in UCSs caused by unclear specification of interactions between the basic flow and its corresponding alternative flows.

Each alternative flow must have a postcondition. It is usual to provide a postcondition describing a constraint that must be true when a use case terminates. If the use case contains alternative flows, then the postcondition of the use case should describe not only what must be true when the basic flow terminates but also what must be true when each alternative flow terminates. The branching condition to each alternative flow is then necessarily part of the postcondition (to distinguish the different possible results). In such a case, the postcondition becomes complex and the branching condition for each alternative flow is redundantly described (both in the steps of flows and the postcondition), which therefore increases the risk of ambiguity in UCSs. Our template enforces that each flow of events (both basic flow and alternative flows) of a UCS contains its own postcondition and therefore avoids such ambiguity.

Table 1 Use case template

Use Case Name	The name of the use case. It usually starts with a verb.	
Brief Description	Summarizes the use case in a short paragraph.	
Precondition	What should be true before the use case is executed.	
Primary Actor	The actor which initiates the use case.	
Secondary Actors	Other actors the system relies on to accomplish the services of the use case.	
Dependency	Include and extend relationships to other use cases.	
Generalization	Generalization relationships to other use cases.	
Basic Flow	Specifies the main successful path, also called "happy path".	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the basic flow executes.
Specific Alternative Flows	Applies to one specific step of the 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.

RUCM Restriction Rules

RUCM contains 26 restriction rules to restrict the way that users write use case specifications. They are based on a thorough literature review and have been experimentally evaluated to be understandable, easy to apply, and not restrictive. Together with the RUCM template, we can expect more precise and comprehensible use case models.

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

Restriction rules on the use of natural language and apply only to action steps (R1-R7)

Seven restriction rules (R1-R7) constrain the use of natural language (Table 2): 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.

Table 2 Restrictions (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.	

Restriction rules on the use of natural language and apply to all sentences (R8-R16)

Nine restriction rules R8-R16 constraining the use of natural language are presented in Table 3. They 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 are specifically to facilitate automated generation of analysis models, though they can also help reduce ambiguity. These two sets of restrictions are thought to be good practice for writing clear and concise UCSs (e.g., [4, 6, 22]) except for R13 and R15. These two rules are proposed in this work 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 document UCSs. A common approach to do so is to use a domain model and glossary (e.g., [16], [5]) as a basis to write UCSs.

Table 3 Restrictions (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; therefore 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 NL 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.

Restriction rules on the use of keywords for specifying control structures (R17-R25) and R26

The remaining ten restriction rules (Table 4) constrain the use of control structures, except R26 that specifies that each basic flow and alternative flow should have its own postcondition. R17 and R18 specify keywords to describe use case dependencies include and extend. Sentences containing the keywords INCLUDE USE CASE and EXTENDED BY USE CASE are referred to as dependency sentences. R19 specifies keyword RFS, which is used in a specific (or bounded) alternative flow to refer to a step number (or a set of step numbers) of a reference flow step 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 is used in basic flows; while ELSE is used in alternative flows.), and 3) IF-THEN-ELSEIF-THEN-ENDIF (everything is in one flow or IF-THEN is used in basic flows; while ELSEIF-THEN-ENDIF is used in alternative flows.). 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 two complex and composite 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 and we reused them with some variation. R22, R24 and R25 are newly proposed in this work for the purpose of making the whole set of restrictions as complete as possible so that flows of events and interactions between the basic flow and the alternatives can be clearly and concisely specified. Applying this set of rules 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.

Table 4 Restrictions (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 their own postconditions.

Example of Use – Withdraw Fund

An example is worth a thousand of words.

An example of use case descriptions documented with RUCM is presented in Table 5 and Table 6. The original design of the use case description is from [9]. We rewrote it by applying RUCM. As show in Table 5 and Table 6, the use case *Withdraw Fund* contains one basic flow, one specific alternative flow, one bounded alternative flow, and one global alternative flow. The specific and bounded alternatives correspond to four basic steps containing the keyword VALIDATES THAT. Notice that this is just one possible definition of the UCS applying our template and restrictions; it is possible to have different but equivalent definitions: for example, using the keyword IF-THEN-ELSE-ELSEIF-ENDIF instead of VALIDATES THAT.

Table 5 Use case Withdraw Fund (part 1)

Use Case Name	Withdraw Fund	
Brief Description	ATM customer withdraws a specific amount of funds from a valid bank account.	
Precondition	The system is idle. The system is displaying a Welcome message.	
Primary Actor	ATM customer	
Secondary Actors	None	
Dependency	INCLUDE USE CASE Validate PIN.	
Generalization	None	
Basic Flow	**Steps**	
	1	INCLUDE USE CASE Validate PIN.
	2	ATM customer selects Withdrawal through the system
	3	ATM customer enters the withdrawal amount through the system.
	4	ATM customer selects the account number through the system.
	5	The system VALIDATES THAT the account number is valid.
	6	The system VALIDATES THAT ATM customer has enough funds in the account.
	7	The system VALIDATES THAT the withdrawal amount does not exceed the daily limit of the account.
	8	The system VALIDATES THAT the ATM has enough funds.
	9	The system dispenses the cash amount.
	10	The system prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance.
	11	The system ejects the ATM card.
	12	The system displays Welcome message.
	Postcondition	ATM customer funds have been withdrawn.

Table 6 Use case Withdraw Fund (part 2)

Bounded Alternative Flows	**RFS Basic Flow 5-7**	
	1	The system displays an apology message MEANWHILE the system ejects the ATM card.
	2	The system shuts down.
	3	ABORT.
	Postcondition	ATM customer funds have not been withdrawn. The system is shut down.
Global Alternative Flows	IF ATM customer enters Cancel THEN	
	1	The system cancels the transaction MEANWHILE the system ejects the ATM card.
	2	ABORT.
	ENDIF	
	Postcondition	ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message.
Specific Alternative Flows	**RFS Basic Flow 8**	
	1	The system displays an apology message MEANWHILE the system ejects the ATM card.
	2	ABORT.
	Postcondition	ATM customer funds have not been withdrawn. The system is idle. The system is displaying a Welcome message.

In the rest of the section, we learn how RUCM restriction rules are applied based on the example by comparing with the original design of the use case *Withdraw Fund*.

PRECONDITION:

Original sentence: ATM is idle, displaying a Welcome message.
Rewritten sentence: The system is idle. The system is displaying a Welcome message.
Reason: The original sentence breaks R15, which suggests not using participle phrases as adverbial modifier. Therefore, the original sentence is split into two simple sentence.

BASIC FLOW STEP 1

Original sentence: Include Validate PIN abstract use case
Rewritten sentence: Basic Flow Step 1
Reason: the keyword INCCLUDE USE CASE specified as R17, should be used to describe the dependency relationship between use cases *Withdraw Fund* and *Validate PIN*.

BASIC FLOW STEP 2-4

Original sentence: Customer selects Withdrawal, enters the amount, and selects the account number.
Rewritten sentences: Basic Flow Step 2, Step 3, and Step 4
Reason: the original sentence is not a simple sentence (violating R12 and R4); it contains three predicates, which implies three action steps. Besides, it is not clear whether these three actions occur concurrently or sequentially. Therefore, we rewrite the sentence into three sequential action steps: steps 2-4.

As specified in R9, the terms should be used in a consistent way; therefore "ATM customer" should be used instead of "Customer: in the original sentence.

R7 says that the interaction between actors and the system should be clearly described without omitting its sender and receiver. The original sentence violates R7 and we rewrite the sentence by explicitly saying "…through the system".

BASIC FLOW STEP 5-9 + ALTERNATIVE FLOWS

Original sentences: Basic flow:

1) System checks whether customer has enough funds in the account and whether the daily limit will not be exceeded.
2) If all checks are successful, system authorizes dispensing of cash.

Alternative flows:

1) If the system determines that the account number is invalid, it displays an error message and ejects the card.
2) If the system determines that there are insufficient funds in the customer's account, it displays an apology and ejects the card.
3) If the system determines that the maximum allowable daily withdrawal amount has been exceeded, it displays an apology and ejects the card.
4) If the ATM is out of funds, the system displays an apology, ejects the card, and shuts down the ATM.

Rewritten sentences: Basic Flow Step 5, Step 6, Step 7, and Step 8

Reason: R22 specifies the keyword VALIDATES THAT, which means that the condition is evaluated by the system and must be true to proceed to the next step. We use this keyword to rewrite all the original sentences. Step 1 of the original basic flow contains two condition checking sentences and the problem is that it is not clear with condition checking action should be taken first or both of them occur concurrently. By using the keyword "VALIDATES THAT", there is no need to have the "if" condition of Step 2 of the original basic flow and therefore the ambiguity is avoided and the description is simplified.

The first original alternative flow contains an "if" condition which should be described as a condition check action sentence in the basic flow (Basic Flow Step 5 of the rewritten use case) by using the keyword VALIDATES THAT. The same rule is applied to the other three alternative flows of the original use case specification.

The alternative flows of the original use case specification do not clearly specify the locations where they branched from the basic flow. The keyword RFS should be applied to clearly state the location (step) that an alternative branches from its reference flow.

It is also important to distinguish different types of alternative flows.

The basic flow and each alternative flow should have its own postcondition.

Each alternative flow must end either with ABORT or RESUME STEP.

Quick Reference Tables

The quick reference tables available here provide a way for users quickly look at a specific restriction rule including its definition, explanation, and example.

Table 7 Restriction rules R1-R7

#	Description	Example	
		RIGHT	**WRONG**
0	A step is either one simple sentence (see below) or one complex sentence (with keywords IF, WHILE, VALIDATES, DO—see below).	N/A	N/A
R1	The subject of a sentence in basic and alternative flows should be the system or an actor.	*The system* ejects the ATM card	*The card* has been ejected
R2	Describe the flow of events sequentially.	1. The system dispenses the cash amount. 2. The system ejects the ATM card.	1. The system ejects the ATM card. 2. The system dispenses the cash amount.
R3	Actor-to-actor interactions are not allowed.	The customer inserts the ATM card into the card reader.	The customer gives the teller the ATM card.
R4	Describe one action per sentence. (Avoid compound predicates.)	the system cancels the transaction MEANWHILE the system ejects the card. or 1. the system cancels the transaction. 2. the system ejects the card	…the system cancels the transaction and ejects the card.
R5	Use present tense only.	The system *ejects* the card.	The system *ejected* the card.
R6	Use active voice rather than passive voice.	The system *ejects* the card.	The card *is ejected*.
R7	Clearly describe the interaction between the system and actors without omitting its sender and receiver.	ATM customer enters PIN number *to the system*.	Customer enters PIN.

Table 8 Restriction rules R8-R16

#	Description	Example	
		RIGHT	**WRONG**
R8	Use declarative sentence only. "Is the system idle?" is a non-declarative sentence.	The system ejects the card.	Ejects the card.
R9	Use words in a consistent way.	*ATM customer* inserts the ATM card…	*Customer* inserts the ATM card…
R10	Don't use modal verbs (e.g., *might*)	The system ejects the card.	The system *might* eject the card.
R11	Avoid adverbs (e.g., *very*).	The system ejects the card.	The system *very likely* ejects the card.
R12	Use simple sentences only. A simple sentence must contain only one subject and one predicate.	1. The system displays ATM customer accounts. 2. The system prompts ATM customer for …	System displays customer accounts *and* prompts customer for transaction type…
R13	Don't use negative adverb and adjective (e.g., *hardly*, *never*), but it is allowed to use *not* or *no*.	The PIN number *has not* been validated	The PIN number *has never been* validated.
R14	Don't use pronouns (e.g. *he*, *this*)	…*the system* reads the card number.	…*it* reads the card number.
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.	The system is idle. The system is displaying a Welcome message.	ATM is idle, displaying a Welcome message.
R16	Use declarative sentence only. "Is the system idle?" is a non-declarative sentence.	the system	"ATM" or "The ATM system"

Table 9 Restriction rules R17-R20

#	Description	Grammar	Explanation	Example	
				RIGHT	WRONG
R17	Use keywords INCLUDE USE CASE to describe the include dependencies with other use cases.	INCLUDE USE CASE <included use case name>	The keywords can be used in basic and step alternative flows.	INCLUDE USE CASE Validate PIN	Include Validate PIN abstract use case.
R18	Use keywords EXTENDED BY USE CASE to refer to the extended use case.	EXTENDED BY USE CASE <extending use case> or <specific use case> Appears either in the action part of a THEN or ELSE in the main flow, or as an action in an alternative flow.	The keywords can be used in basic and alternative flows.	EXTENDED BY USE CASE CreateIncident	Use case CreateIncident extends the current use case.
R19	Use keyword RFS in a specific (or bounded) alternative flow to refer to a step number (or a lower bound step number and an upper bound step number) of a reference flow step that this alternative flow corresponds to.	RFS <reference flow step #> (specific alternative flow) RFS <reference flow step numbers> (bounded alternative flow) Not required notation for global alternative flow.	One specific or bounded alternative flow must correspond to exactly one or more than one reference flow steps.	RFS Basic Flow 5 … RFS Basic Flow 5-7, 10, 14 …	
R20	Use pairs of keywords of IF, THEN, ELSE, ELSEIF, and ENDIF to describe conditional logic sentences.	IF <condition> THEN <steps> ENDIF	Appears in one flow only.	IF the system recognizes the ATM card, THEN the system reads the ATM card number, ENDIF.	If the system recognizes the card, it reads the card number.
		IF <condition> THEN <steps> – ELSE <steps> ENDIF	IF-THEN is used in basic flows; while ELSE is used in alternative flows (see RFS). Or everything is used in only one flow.		
		IF <condition> THEN <steps> – ELSEIF <condition> THEN <steps> ENDIF	IF-THEN is used in basic flows; while ELSEIF-THEN is used in alternative flows (see RFS). Or everything is used in only one flow.		

Table 10 Restriction rules R21-R26

#	Description	Grammar	Explanation	Example	
				RIGHT	WRONG
R21	Use keyword MEANWHILE to describe concurrency.	<action> MEANWHILE <action>	It implies that the sentence before keyword MEANWHILE and the sentence after the keyword occur concurrently.	…the system cancels the transaction MEANWHILE the system ejects the card.	…the system cancels the transaction and ejects the card.
R22	Use keyword VALIDATES THAT to describe condition checking sentences. VALIDATES THAT means that the condition is evaluated and must be true to proceed to the next step.	VALIDATES THAT <condition>	The alternative case (the condition does not hold) must be described in its corresponding alternative flow (BFS).	…the system VALIDATES THAT the user- entered PIN…	… the system checks whether the user-entered PIN…
R23	Use keyword pair DO and UNTIL to describe iteration.	DO <steps> UNTIL <condition >	Following keyword DO is a sequence of steps. Following keyword UNTIL is a loop ending condition.	1. DO 2. action1 3. action2 4. UNTIL condition	
R24	Use keyword ABORT to describe an exceptionally exit action. An alternative flow ends either with ABORT or RESUME STEP.	ABORT	Used in alternative flows, iterative, and conditional logic sentences. It means the ending of a use case.		
R25	Use keyword pair RESUME STEP to describe the situation where an alternative flow goes back to its corresponding basic flow. An alternative flow ends either with ABORT or RESUME STEP.	RESUME STEP <basic flow step #>	Used in alternative flows.	RESUME STEP 5	
R26	Each basic flow and alternative flow should have their own postconditions.	Refer to restriction R19.			