securityandtrust.lu

Restricted Use Case Modeling (RUCM)
User Manual v2.1

Software Verification and Validation Laboratory
Interdisciplinary Centre for Security, Reliability and Trust
University of Luxembourg

March 25, 2016

UNIVERSITÉ DU LUXEMBOURG

Contents

1 Introduction

Use case modeling, including use case diagrams and textual specifications, is commonly used to structure and document requirements. Use Case Specifications (UCS) are textual documents, formatted according to a use case template, specifying use cases. Restricted Use Case Modeling (RUCM) is a use case modeling approach which is composed of a set of well-defined language and grammar restriction rules, along with a use case template. The goal of RUCM is to restrict the way users specify UCSs to reduce ambiguity, improve understandability, and ease the construction of analysis models either manually or automatically. The original work on RUCM was conducted by Tao Yue under the supervision of Prof. Lionel Briand. This document is a modified version of the original user manual (2010).

In the rest of this user manual, we first describe the RUCM use case template and the RUCM restriction rules. Then through a running example we illustrate how to specify use case models with RUCM.

2 RUCM Use Case Template

The RUCM use case template is based on related work about use case modeling. It contains fields that are commonly encountered in conventional templates, but better specifies the event flow structure of a use case specification. The 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 in Table 1).

Use Case Name Gives the name of the use case. It usually starts with a verb (e.g., *Identify Initial Occupancy Status*) and should be consistent with the name of the same use case in the use case diagram.

Brief Description Summarizes the use case in a short paragraph. It captures the essence of the use case.

Precondition Specifies what must be always true before the use case begins.

Primary Actor The principle actor which initiates the use case.

Secondary Actors Actors that the system relies on to accomplish the use case.

Dependency Specifies «include» and «extend» relationships of the use case to other use cases.

Generalization Specifies the generalization relationship of the use case to other use cases.

Basic Flow Describes a main successful path that satisfies stakeholder interests. It typically does not include any condition or branches. It is recommended to describe conditions and branches in alternative flows separately. A basic flow is composed of a sequence of steps and a postcondition. Each UCS can only have one basic flow.

The use case steps can be one of the following type of steps:

1) **Primary Actor** $\rightarrow$ **System** A primary actor sends a request and/or data to the system (see R27 and R28 in Table 4);

2) **Validation** The system validates a request and/or data (see R22 in Table 4);
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 (see R27 in Table 4);
5) **System → Secondary Actor:** The system sends a request to a secondary actor (see R28 in Table 4).

All steps are numbered sequentially. Each step is completed before the next one is started. If there is a need to express conditions, iterations, or concurrency, then specific keywords, specified as restriction rules in Section 3, 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 the alternative flow. The branching condition is specified in the reference flow by following restriction rules (see R20 and R22 in Table 4). 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 an alternative flow that refers to more than one step in the reference flow - consecutive steps or not.
3) A *global alternative flow* 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, outline in rule R19 (see Table 4), is used to specify one or more reference flow step numbers. Whether the control flow returns to the original flow (RESUME STEP), or terminates the use case (ABORT), must be specified as the last step of the alternative flow.

Similar to the branching condition, merging and termination are specified by following restriction rules (see R24 and R25 in Table 4). By doing so, we can avoid potential ambiguity in UCSs, caused by unclear specification of interactions between the basic flow and its corresponding alternative flows. Each alternative flow must have a postcondition (enforced by the restriction rule R26 in Table 4).

Typically, a postcondition should describe 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 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 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.

Table 1: 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 is executed.
Specific Alternative Flow	Applies to one specific step of the basic flow.	
	RFS	A reference flow step number where the basic flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow is executed.
Global Alternative Flow	Applies to all the steps of the basic flow.	
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow is executed.
Bounded Alternative Flow	Applies to more than one step of the basic flow, but not all of them.	
	RFS	A list of reference flow steps where the basic flow branches from.
	Steps (numbered)	Flow of events.
	Postcondition	What should be true after the alternative flow is executed.

The alternative flows can be named. e.g., 'Specific Alternative Flow - A' and 'Bounded Alternative Flow - 1'. Naming is optional but useful when there is an alternative flow referring to another alternative flow. For instance, a specific alternative flow can have a condition step which a second specific alternative flow refers to. The first alternative flow can be named as 'Specific Alternative Flow - <*name*>' and the second alternative flow can refer to the condition step in the first alternative flow with the RSF section as 'RFS - SAF<*name*>- <*reference flow step number*>' where SAF stands for Specific Alternative Flow. Name of an alternative flow can be any character, string or number.

Before the execution of a reference flow step, global alternative flows are always taken first to check if their conditions hold. Then, conditions of bounded alternative flows referring to the reference flow step are checked. After that, the reference flow step is taken into account. If the reference flow step is a condition step, its condition is checked. If the condition does not hold, specific alternative flows referring to the reference flow step are taken to check their conditions. If there are multiple alternative flows of the same type referring to the same reference flow step, their conditions are checked according to the order in which they are specified in the specification.

3 RUCM Restriction Rules

The 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 and actor-system interactions. The first group 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 applied only to action steps (R1-R7)

Seven restriction rules (R1-R7) constrain the use of natural language (see Table 2). The second column in Table 2 explains why these rules are needed in order 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: 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.	

Restriction rules on the use of natural language applied 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 specifically facilitate automated generation of analysis models, e.g., state machines, though they can also help reduce ambiguity. Rules R8-R16 are thought to be good practice for writing clear and concise UCSs. R13 and R15 are proposed because 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 a glossary as a basis to write UCSs.

Table 3: Restriction Rules R8-R16

#	Description	Explanation
R8	Use declarative sentence only. For example, "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 and actor-system interactions (R17-R28)

The remaining twelve restriction rules (see Table 4) constrain the use of control structures and system-actor interactions, 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 the 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. The reference flow can be Basic Flow, Specific Alternative Flow (SAF), Bounded Alternative Flow (BAF) or Global Alternative Flow (GAF).

Rules R20-R23 give the keywords in order to specify conditional logic sentences (IF-THEN-ENDIF), concurrency sentences (MEANWHILE), condition checking sentences (VALIDATES THAT), and iteration sentences (DO-UNTIL), respectively.

The keyword IF-THEN-ENDIF is used only in alternative flows for conditions that need to be specified in the beginning of the alternative flow. When such a condition in an alternative flow is given with IF-THEN, the rest of the steps in the alternative flow should be given between IF-THEN and ENDIF. Global and bounded alternative flows should always be specified within IF-THEN-ENDIF. If the specific alternative flow is the last one referring to the same reference flow step or the only alternative flow referring to the same condition step in the reference flow, it should not contain IF-THEN-ENDIF.

Table 4: Restriction Rules R17-R26

#	**Description**	#	**Description**
R17	INCLUDE USE CASE	R23	DO-UNTIL
R18	EXTENDED BY USE CASE	R24	ABORT
R19	RFS	R25	RESUME STEP
R20	IF-THEN-ENDIF	R26	Each basic flow and alternative flow should have its own postconditions.
R21	MEANWHILE	R27	SENDS-TO
R22	VALIDATES THAT	R28	REQUESTS-FROM

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. According to these two rules, an alternative flow ends either with ABORT or RESUME STEP. With RESUME STEP followed by a returning step number, the execution returns back to the reference flow from the alternative flow. The execution terminates with ABORT.

R17-R21 and R23 are adopted from the previous literature with some variations. R22, R24 and R25 are newly proposed 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., one can correctly parse sentences with our specified keywords.

R27 and R28 introduce the keywords 'SENDS .. TO' and 'REQUESTS .. FROM' to distinguish system-actor interactions. The system/actor sends a message/data to the actor/system without any request, or the system/actor requests a message/data from the actor/system and in return the requested message/data is sent.

4 Example Use Case Specifications

An example use case specification documented with RUCM is presented in Table 5 and Table 6. In constructing this, we examined a use case specification modeled with a conventional use case modeling technique and rewrote it using RUCM. As shown in Table 5 and Table 6, the use case *Identify Initial Occupancy Status* contains one basic flow, one bounded alternative flow, and three specific alternative flows.

In the following, we show how RUCM restriction rules are applied based on the example, comparing the RUCM version against the original use case.

Table 5: Use case Identify Initial Occupancy Status (part 1)

Use Case Name	Identify Initial Occupancy Status	
Brief Description	The system notifies the airbag control unit with the initial occupancy status.	
Precondition	The system is ready to handle the main functionality.	
Primary Actors	IgnitionResetButton	
Secondary Actors	AirbagControlUnit	
Dependency	INCLUDE USE CASE Initialize System for Normal Operation, ...	
Generalization	None	
Basic Flow	**Steps**	
	1	INCLUDE USE CASE Initialize System for Normal Operation.
	2	INCLUDE USE CASE Self Diagnose System.
	3	INCLUDE USE CASE Classify Occupancy Status.
	4	The system VALIDATES THAT the occupant classes for airbag control and seat belt reminder are valid.
	5	The system SENDS the occupant class for airbag control TO AirbagControlUnit.
	6	The system SENDS the occupant class for seat belt reminder TO AirbagControlUnit.
	Postcondition	The occupant classes for airbag control and seat belt reminder have been sent.

BASIC FLOW STEPS 1-3

Original sentence: Include Initialize system for normal operation.

Rewritten sentence: Basic Flow Step 1

Reason: The keyword INCLUDE USE CASE specified in R17 should be used to describe the dependency relationship between use cases *Identify Initial Occupancy Status* and *Initialize System for Normal Operation*. R17 should also be applied for other inclusion of use cases in the basic flow (Steps 2 and 3). In addition, all the dependency relations should be first given in the *Dependency* section of the including use case *Identify Initial Occupancy Status*.

BASIC FLOW STEPS 4-6

Original sentence: System validates and provides occupant classes for airbag control and seat belt reminder to airbag controller.

Rewritten sentences: Basic Flow Step 4, Step 5, and Step 6

Reason: The original sentence is not a simple sentence (violating R12 and R4); it contains two predicates which imply two action steps and one condition step. Besides, it is not clear whether these two actions occur concurrently or sequentially. Therefore, we rewrite the sentence as one condition step and two sequential action steps (Steps 4-6).

As specified in R9, the terms should be used in a consistent way. Therefore, "Airbag-ControlUnit" should be used instead of "airbag controller" in the original sentence. Also the system should always be referred as "the system" in the use case steps (R16).

R7 states that the interaction between actors and the system should be clearly described without omitting its sender and receiver. R27 and R28 enforces the use of keywords "SENDS-TO" and "REQUESTS-FROM" for describing the system-actor interactions. The original sentence violates R7, R27 and R28. Therefore, we rewrite the system-actor interactions in the sentence by explicitly saying "The system SENDS ... TO AirbagControlUnit" in two action steps.

Table 6: Use case Identify Initial Occupancy Status (part 2)

Bounded Alternative Flow	**RFS 1 - 4**	
	1	IF the voltage error is detected THEN
	2	The system resets classification filters.
	3	RESUME STEP 2.
	4	ENDIF.
	Postcondition	Classification filters have been reset.
Specific Alternative Flow - A	**RFS 4**	
	1	IF the occupant class for airbag control is not valid and the occupant class for seat belt reminder is not valid THEN
	2	The system SENDS the previous occupant class for airbag control TO AirbagControlUnit.
	3	The system SENDS the previous occupant class for seat belt reminder TO AirbagControlUnit.
	4	The system SENDS the DTC TO AirbagControlUnit.
	5	RESUME STEP 2.
	6	ENDIF.
	Postcondition	The previous occupant classes for airbag control and seat belt reminder have been sent to AirbagControlUnit.
Specific Alternative Flow - B	**RFS 4**	
	1	IF the occupant class for airbag control is not valid THEN
	2	The system SENDS the previous occupant class for airbag control TO AirbagControlUnit.
	3	The system SENDS the occupant class for seat belt reminder TO AirbagControlUnit.
	4	The system SENDS the DTC TO AirbagControlUnit.
	5	RESUME STEP 2.
	6	ENDIF.
	Postcondition	The previous occupant class for airbag control and the occupant class for seat belt reminder have been sent to AirbagControlUnit.
Specific Alternative Flow - C	**RFS 4**	
	1	The system SENDS the occupant class for airbag control TO AirbagControlUnit.
	2	The system SENDS the previous occupant class for seat belt reminder TO AirbagControlUnit.
	3	The system SENDS the DTC TO AirbagControlUnit.
	4	RESUME STEP 2.
	Postcondition	The occupant class for airbag control and the previous occupant class for seat belt reminder have been sent to AirbagControlUnit.

BASIC FLOW STEP 4 + ALTERNATIVE FLOWS

Original sentences:

Basic flow:

1) System validates and provides occupant classes for airbag control and seat belt reminder to airbag controller.

Alternative flows:

1) If the system detects a voltage error, it resets the classification filters.
2) If the occupant classes for airbag control and seat belt reminder are not valid, it sends the previous occupant classes for airbag control and seat belt reminder.
3) If the occupant class for airbag control is not valid and the occupant class for seat belt reminder is valid, it sends the previous occupant class for airbag control and the occupant class for seat belt reminder.
4) If the occupant class for airbag control is valid and the occupant class for seat belt reminder is not valid, it sends the occupant class for airbag control and the previous occupant class for seat belt reminder.

Rewritten sentences: Basic Flow Step 4, Step 5, Step 6 and Alternative Flows
Reason: R22 specifies the keyword *VALIDATES THAT* which means the condition is evaluated by the system and must be true to proceed with the next step. We used this keyword to rewrite the condition in the basic flow step. The original step of the basic flow contains a condition statement which does not clearly describe what the exact condition is. By using the keyword *VALIDATES THAT* in RUCM, we explicitly specify that the occupant classes for both airbag control and seat belt reminder should be valid to proceed with the next step in the basic flow.

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

It is also important to distinguish different types of alternative flows. The basic flow and its alternative flows should have their own postconditions (see postconditions in Table 5 and Table 6). Each alternative flow must end either with *ABORT* or *RESUME STEP* (see Table 6). Alternative flows in RUCM is named in the form of *Alternative Flow - <flow name>*, e.g., *Specific Alternative Flow - A* in Table 6. As mentioned before, naming is optional and name of the alternative flow can be any character, string or number. However, it is recommended to use sequential characters (e.g., *A*, *B*, *C*) and numbers (e.g., 1, 2, 3, 4) for naming alternative flows.

The original alternative flows contain "if" conditions which should be described as a condition before the steps in the alternative flows by using the keyword *IF-THEN*. The execution order of alternative flows in RUCM depends on the type of alternative flows and the order in which they are specified. In Table 6, the *IF* condition of the bounded alternative flow is checked first. If it does not hold, the conditions in the *IF-THEN* statements of the specific alternative flows referring to the same condition step in the reference flow are checked based on the order in which the alternative flows are specified. Please note that if the condition step in the reference flow is referred by only one alternative flow, that alternative should not have an *IF-THEN* statement to ensure that the alternative flow is taken when the condition does not hold. If there are multiple specific alternative flows referring to the same condition step in the reference flow, the last alternative flow cannot have an *IF-THEN* statement. That is why the last specific alternative flow in Table 6 does not start with an *IF-THEN* statement. It is ensured that the last specific alternative flow is executed if the conditions of all other alternative flows do not hold. If an alternative flow starts with an *IF-THEN* statement, all other steps including *ABORT* and *RESUME STEP* should be specified in between the *IF-THEN* and *ENDIF* statements.

As given in Table 7, alternative flows can be named. The names are used when there is an alternative flow referring to a reference step in another alternative flow. Table 7 gives two alternative flows for the use case Qualify and Dequalify Errors.

Table 7: Some Alternative Flows for the Use Case Qualify and Dequalify Errors

Specific Alternative Flow - X	**RFS 2**	
	1	The system qualifies errors with a higher priority than qualified errors
	2	The system VALIDATES THAT no error with discard attribute is qualified.
	3	The system increases the qualification counter for each new qualified error.
	4	RESUME STEP 3.
	5	ENDIF.
	Postcondition	Errors have been qualified with a higher priority. The qualification counter for each new qualified error have been increased.
Specific Alternative Flow	**RFS SAFX-2**	
	1	The system sets DiscardFlag.
	2	The system sets DiscardError as detected.
	3	The system sets DiscardError as qualified.
	4	RESUME STEP SAFX-2.
	Postcondition	The system has set DiscardFlag. The system has set DiscardError.

The first specific alternative flow is named as *X*. It contains a condition step referred by the second specific alternative flow. After the keyword *RFS*, we use the name of the alternative flow of the reference flow step and the step number together with *SAF*, i.e., *RFS SAFX-2*, while *SAF* stands for Specific Alternative Flow.

5 Quick Reference Table

(See next pages)

Table 8: Restriction Rules (R1-R16)

#	Description	Example	
		RIGHT	WRONG
R1	The subject of a sentence in basic and alternative flows should be the system or an actor.	The system sets DiscardFlag	DiscardFlag has been set
R2	Describe the flow of events sequentially.	1. The system sets error as detected. 2. The system SENDS error message TO AirbagControlUnit.	1. The system SENDS error message TO AirbagControlUnit. 2. The system sets error as detected.
R3	Actor-to-actor interactions are not allowed.	The system SENDS message TO AirbagControlUnit.	SeatBeltReminder SENDS message TO AirbagControlUnit.
R4	Describe one action per sentence. (Avoid compound predicates.)	1. The system updates error traceability buffer. 2.The system resets ignition counter.	The system updates error traceability buffer and resets ignition counter.
R5	Use present tense only.	The system *maps* QualifiedErrors to DTC.	The system *mapped* QualifiedErrors to DTC.
R6	Use active voice rather than passive voice.	The system *maps* QualifiedErrors to DTC.	QualifiedErrors *are mapped* to DTC.
R7	Clearly describe the interaction between the system and actors without omitting its sender and receiver.	The system SENDS class TO *AirbagControl*.	The system SENDS class.
R8	Use declarative sentence only. "Is the system idle?" is a non-declarative sentence.	The system sets DiscardFlag.	Set DiscardFlag.
R9	Use words in a consistent way.	The system SENDS class TO *AirbagControlUnit*	The system SENDS class TO *airbag*
R10	Do not use modal verbs (e.g., *might*)	The system sets DiscardFlag.	The system *might* set DiscardFlag.
R11	Avoid adverbs (e.g., *very*).	The system sets DiscardFlag.	The system *very likely* sets DiscardFlag.
R12	Use simple sentences only. A simple sentence must contain only one subject and one predicate.	1.The system sets errors as *not* detected. 2. The system sets DiscardFlag	System sets errors as not detected *and* sets DiscardFlag...
R13	Do not use negative adverb and adjective (e.g., *hardly, never*), but it is allowed to use *not* or *no*.	Errors have *not* been qualified (postcondition)	Errors have *never* been qualified (postcondition)
R14	Do not use pronouns (e.g. *he, this*)	...*the system* sets DiscardFlag.	...*it* sets DiscardFlag.
R15	Do not use participle phrases as adverbial modifier. For example, the italic-font part of the sentence "The system is idle, *waiting for ignition message*" is a participle phrase.	The system is idle. The system is waiting for ignition (precondition)	The system is idle, waiting for ignition (precondition)
R16	Use "*the system*" to refer to the system under design consistently	the system	"BodySense" or "The BodySense 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 alternative flows.	INCLUDE USE CASE Self Diagnose System	Include Self Diagnose System 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 as an action in the basic flow or in an alternative flow.	The keywords can be used in basic and alternative flows.	EXTENDED BY USE CASE Create Incident	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 <step # > or RFS SAF<SAF name > - <SAF step # > or RFS BAF<BAF name > - <BAF step # > or RFS GAF<GAF name > - <GAF step # >	One specific or bounded alternative flow must correspond to exactly one or more than one reference flow steps.	RFS 5 ... RFS 5-7, 10, 14 RFS SAF A-1 RFS BAF 1-4 RFS GAF Y-3 ...	
R20	Use pairs of keywords of IF, THEN, and ENDIF to describe conditional logic sentences in alternative flows.		If the alternative flow is the last one referring to the same reference flow step or the only alternative flow referring to the same condition step in the reference flow, it does not contain condition sentence.		

		IF <condition>THEN
		<steps>
		ABORT
		ENDIF
			If the use case is terminated, ENDIF should always be just after ABORT.		

		IF <condition>THEN
		<steps>
		RESUME STEP #
		ENDIF
			If the use case is not terminated, ENDIF should always be just after RESUME STEP.		

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 the keyword 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.	...the system VALIDATES THAT class is valid.	the system checks if class is valid
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 reference flow. An alternative flow ends either with ABORT or RESUME STEP.	RESUME STEP <step # > or RESUME STEP SAF <SAF name >- <SAF step # > or RESUME STEP BAF <BAF name >- <BAF step # > or RESUME STEP GAF <GAF name >- <GAF step # >	Used in alternative flows. The execution returns to the original flow step given with the step number in RESUME STEP.	RESUME STEP 5 RESUME STEP SAFA-1 RESUME STEP BAF3-1 RESUME STEP GAFX-5	
R26	Each basic flow and alternative flow should have their own postconditions.	Refer to restriction R19.			
R27	Use keyword SENDS-TO to describe system-actor interaction.	<system>| <actor> SENDS <data>TO <actor>| <system>	Actor-to-actor interaction is not allowed.	The system SENDS class TO AirbagControlUnit	
R28	Use keyword REQUESTS-FROM to describe system-actor interaction.	<system>| <actor> REQUESTS <data>FROM <actor>| <system>	Actor-to-actor interaction is not allowed.	AirbagControlUnit REQUESTS class FROM the system	