OETCS

ITEA 2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

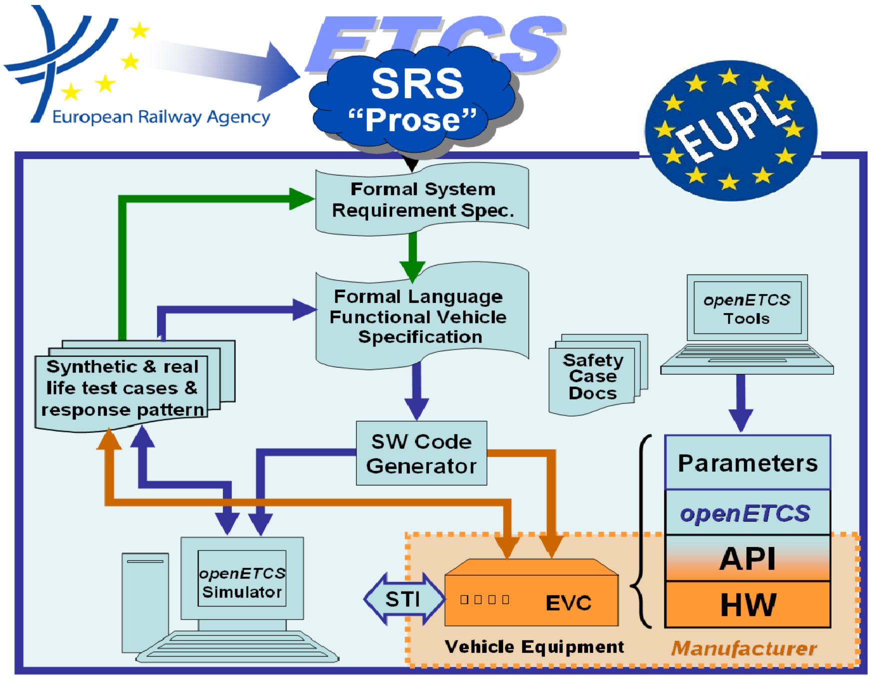
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Toolchain”

Radio Communication Management
SRS SUBSET-026-3.5
SysML Model

Cécile Braunstein

June 2013



supported by:
Federal Ministry of Education and Research
Liberté • Égalité • Fraternité
République Française
Ministère de l'Enseignement Supérieur et de la Recherche
Investissements d'Avenir
Région de Bruxelles-Capitale
Gobierno de España
Ministerio de Industria, Energía y Turismo

openETCS

This page is intentionally left blank

Work-Package 7: “Toolchain”

OETCS
June 2013

Radio Communication Management
SRS SUBSET-026-3.5
SysML Model

Cécile Braunstein

Bremen University

Model Description

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This document is a part of the model-evaluation project of WP7, it explores SysML capabilities to provide a (semi)formal model of the SRS SUBSET-026.

Disclaimer: This work is licensed under the European Union Public Licence (EUPL v.1.1) and a Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

Table of Contents

1 Short Introduction to Formalism and Tool 5
2 Modeling Strategy 5
3 Model Overview 6
3.1 Block and state chart view 6
3.2 Requirement view 8
4 Model Benefits 9
5 Detailed Model Description 10
5.1 Abstraction 10
5.2 Inputs and outputs messages 10
5.3 Internal variables 13
5.4 Behaviour 13
5.5 Test cases generation 16
References 17

Figures and Tables

Figures

Figure 1. Methodology example 6
Figure 2. Interactions between the MoRC and others On-board modules 7
Figure 3. Model Overview 7
Figure 4. SUT Overview 8
Figure 5. Example of modeling a requirement 8
Figure 6. Radio control manager interfaces. 11
Figure 7. Automaton of the radio communication management 14
Figure 8. Automaton of the network registration 15

Tables

Table 1. Messages exchange during the Management of Radio Communication 11
Table 2. Test cases generation summary 17

1 Short Introduction to Formalism and Tool

This document describes the modeling of the radio communication management. The model has been made from the specification description of the SRS SUBSET-026-3.5 baseline 3 as recommended by D2.5 Methods and tools benchmarking methodology [3].

The formalism used is SysML [4]. More particularly we had used block diagram, state charts and requirements diagram. SysML is a graphical language that extends UML for a customize version suitable for system engineering. It may help modeling system within a board range of system variety that may include hardware, software, data, personnel and facilities. It supports the specification, analysis design, verification and validation of complex system.

Enterprise architect (EA) version 9.3 [9] has been used to implement the model. Note that this tool is not open source but others tools such as Papyrus [2] provides SysML modeling capabilities and are evaluate by others partners. EA is a visual platform for designing and constructing software systems, for business process modeling, and for more generalized modeling purposes. it covers all aspects of the development cycle. The main advantages is the requirement management and tracing, the team work and the include versionning. The main cons : it is not an open source tool.

We had design a test model, this model aims at generate test cases and test data. The test generator used here is the tool-suite RT-Tester Model-Based generator (RTT-MBT) [6, 7]. It provides sequences of input data with timing constraints that are used for the stimulation of the system under test (SUT), concurrently with generated test oracles.

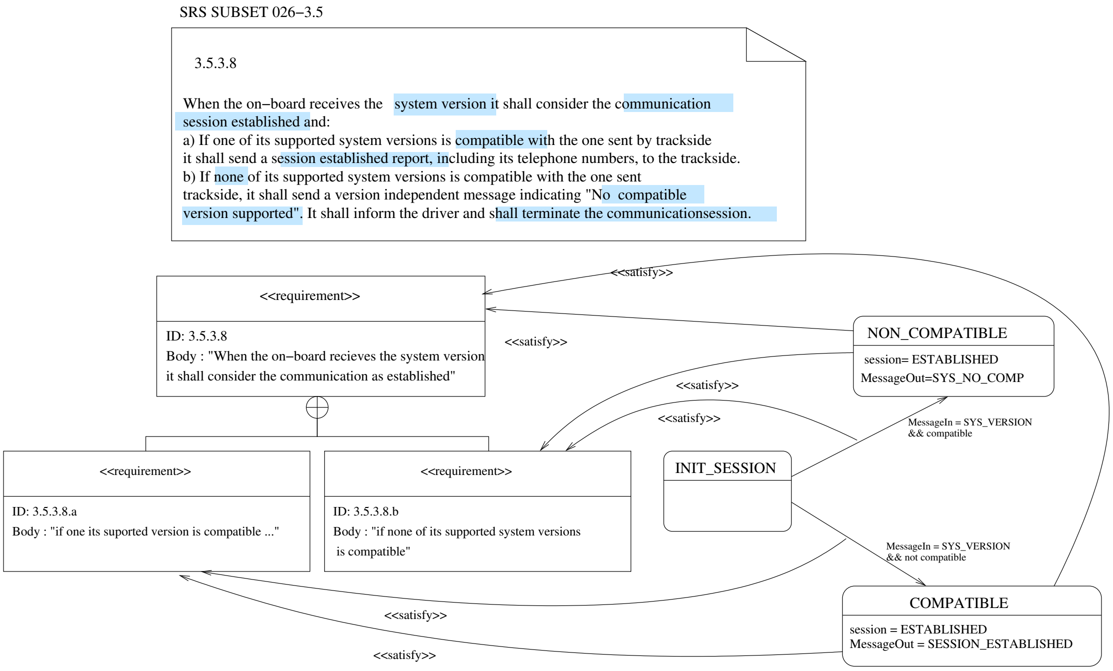
2 Modeling Strategy

Starting from the specification SUBSET-026-chap3.5, we come up with the list of requirements. The requirements are exactly the text of the SRS. This sub-part of the specification defines the management of the radio communication, e.g. the protocol to follow in order to initiate, maintain and terminate a radio communication. From the requirements list, we modeled the set of behaviors defined by direct translation into a state chart diagram. In parallel, we implemented a requirement diagram that links the requirement list and the SRS to the model.

An example figure 1 shows an overview of the translation of the specification into a SysML model. The rest of the document will give more details on each object presented. The basic idea is starting from the textual representation, we can deduce:

- Input events : "receives the system version"
  `MessageIn = SYS_VERSION`
- Internal variables and constants : "The communication session shall be considered as established"
  `session == ESTABLISHED`
- Outputs : "shall send a session established report"
  `MessageOut = SESSION_ESTABLISHED`.

The radio communication management task is part of the set of on-board functions (see SUBSET-026.4.5). The behavior of this function depends on the result of others functions such as the function that determine of the ETCS Mode or Level. This leads us to define an interface between this task and the other tasks of the on board unit (see section 5).



SRS SUBSET 026−3.5

3.5.3.8

When the on−board receives the system version it shall consider the communication session established and:
a) If one of its supported system versions is compatible with the one sent by trackside it shall send a session established report, including its telephone numbers, to the trackside.
b) If none of its supported system versions is compatible with the one sent trackside, it shall send a version independent message indicating "No compatible version supported". It shall inform the driver and shall terminate the communicationsession.

<<requirement>>
ID: 3.5.3.8
Body : "When the on−board recieves the system version it shall consider the communication as established"
<<requirement>>
ID: 3.5.3.8.a
Body : "if one its suported version is compatible ..."
<<requirement>>
ID: 3.5.3.8.b
Body : "if none of its supported system versions is compatible"
<<satisfy>>
<<satisfy>>
<<satisfy>>
<<satisfy>>
<<satisfy>>
<<satisfy>>
NON_COMPATIBLE
session= ESTABLISHED
MessageOut=SYS_NO_COMP
MessageIn = SYS_VERSION && compatible
INIT_SESSION
MessageIn = SYS_VERSION && not compatible
COMPATIBLE
session = ESTABLISHED
MessageOut = SESSION_ESTABLISHED

Figure 1. Methodology example

Moreover the model is designed for test generation purpose. The model of the system under test (SUT) has been made together with the test execution environment (TE). Here the test environment is really trivial since it is empty, it allows any possible value for the inputs of the SUT. One may want to add some constrains on the input signals or want to sketch some behavior such as an event never happened before an other one. This may also be model as a state machine.

3 Model Overview

The model is composed of two packages : the *system* and the *requirement*. The system package consists of the SUT model, the TE model and a set of constants and enumeration types. The requirement package contains all the requirement and a requirement diagram.

3.1 Block and state chart view

The MoRC is the part of the EVC (European vital computer) responsible for the management of radio communication. According to the specification this module interacts directly with the following on-board modules, as shown in figure 2:

- (DMI) driver module interface : receives/displays information from/to the driver,
- (RTM) radio transmission module : receives/gives commands from/to the radio network,
- (BTM) balise transmission module : sends transmission request from a balise group,
- (JRU) Juridical recorder unit : records part of the data exchange for
- (OBU tasks) other on-board functions (SUBSET-026-4.5) used by the MoRC such as:
  - track conditions management functions (SUBSET-026-3.12.1) that determine for instance if the ETCS system of the track is compatible with the system on-board.
  - Mode determination (SUBSET-026-3.12.4, SUBSET-026-4.6).

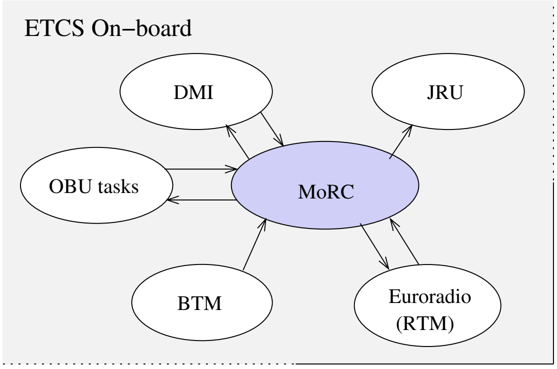


Figure 2. Interactions between the MoRC and others On-board modules

The orders to initiate or terminates a radio communication may come from the DMI, the BTM, and the RBC (radio block centre) through the RTM. The others OBU tasks may also order a radio communication (see section 5.1 for more details).

In figure 3 the MoRC model - *our test model* - is composed with a the test environment (TE). The inputs and outputs interfaces are explained in details in section 5.2. The test environment abstracts away all the others functions or blocks which the SUT may interact with. In our description the test environment is empty, this means that all possible behavior of the environment may be considered.

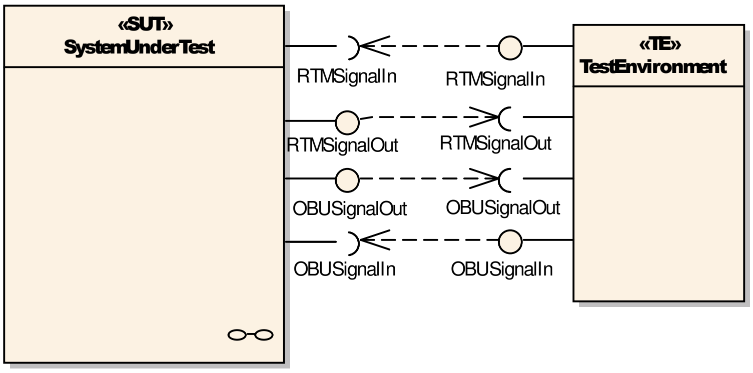


Figure 3. Model Overview

The SUT block consists of two blocks. The SUBSET-026-3.5 specification separates the management of the radio communication into 3 functions :

- The session management from chapter 3.5.3 to 3.5.5
- The registration to the radio network chapter 3.6
- The safe radio connection indication chapter 3.5.7

For the moment we have modeled the first two items. The MoRC block and the RadioNetworkRegistration contains the internal variables used for the protocol implementation and the state-charts describing their behavior. The two state machines are running in parallel. The two objects interact with each other as shown figure 4 and explain later in this document.

Until now, the constants used in this chapter and defined in SUBSET-026.3-A.3.1 are not part of the system under test block. At this stage of modelisation it is not defined where they should be declared and recorded to keep track of possible change or extension. I think they should be part of a special package regrouping all this kind of information. In the current model they belong to the system package.

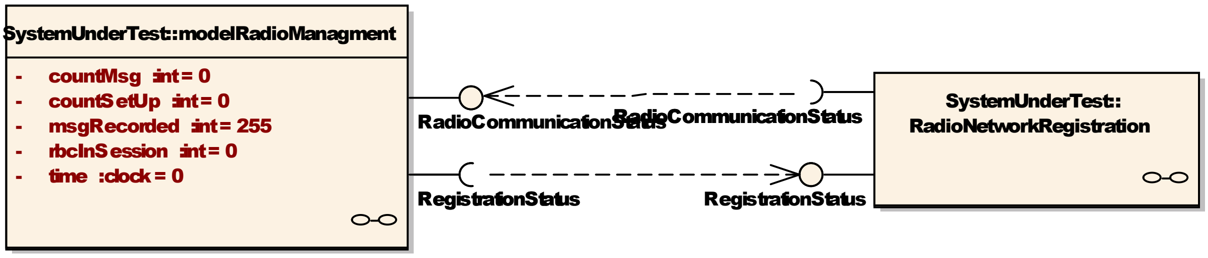


Figure 4. SUT Overview

3.2 Requirement view

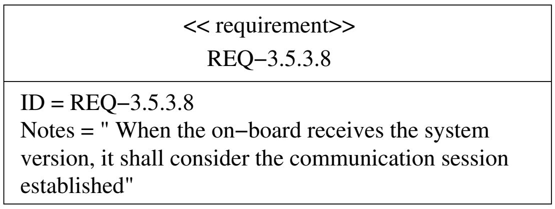
SysML offers a modeling constructs to represent text-based requirements and relationship with other modeling elements. The requirements in EA diagram may be described in a graphical or tree structured format. The requirement may also appear directly other diagram. The diagram requirements intent to may be imported and exported from other tools in csv or XMI format.

In this particular experiment, a table document has been made in csv format from the SRS. Each requirement is represented as a number that corresponds to the numbers and bullet points of the specification document. From this list of requirements, one can directly find the corresponding paragraph in the documents. We had also added the complete text in a text attribute for helping the reader.

The requirement list is then automatically translate to a requirement package in SysML where each requirement is a *requirement* SysML element. Our requirement contains the following attributes :

- ID : the corresponding name from the SRS
- Body: the corresponding text from the SRS

Figure 5 shows an example of the requirement representation.



<< requirement>>
REQ−3.5.3.8

ID = REQ−3.5.3.8
Notes = " When the on−board receives the system version, it shall consider the communication session established"

Figure 5. Example of modeling a requirement

Most of requirement refers to the behaviors model as transition of the state-chart representation. The link has been made such that each transition satisfy at least one requirement. Note that in EA, it is not possible to directly link a transition element to a requirement element, a invariant constraint true has been added on the transition to make the link.

Others requirement may describe a set of transitions or a more general behavior property. In these cases the requirement may be translated as one or more constraints that must be satisfied by the model. In our model, constraints are invariants expressed in LTL[8]. SysML does not imposed any language, one can choose other constraints language such as OCL [5]. The following example shows the translation of two requirements into a LTL properties. The reader should refer to section 5 for variables and values details.

REQ-3.5.3.1 "It shall be possible for ERTMS/ETCS on-board equipment and RBC to initiate a communication session".

```
Finally ((orderOnBoard == 1 || MessageIn == INIT:_SESSION_ORDER ||
MessageIn == INIT_SESSION_TRACK  ) &&
Finally (MessageOut == SESSION_ESTABLISHED))
```

REQ-3.5.3.10 "If the establishment of a communication session is initiated by the RBC, it shall be performed according to the following steps ..."

```
Finally (MessageIn == INIT_SESSION_TRACK && setUp == 1 ->
Next (MessageOut == SESSION_ESTABLISHED && radioComSession == 1ESTABLISHED)
```

The requirement diagram only represent the "satisfy" relations between the requirements and the model. One could refine this diagram and add derive dependency or containment relationship. Note that in practice we could show the "satisfy" relations directly on the state-chart view. The separate representation is more readable.

4 Model Benefits

In the previous sections, we had describes the use of SysML for modeling SUBSET-026.3.5. Note that other aspect of the SRS as define in Deliverable D2.5 may also easily represent with SysML. The study here shows the modelisation of state charts and timeout. Moreover the arithmetic may be represented as parametric diagram with constraints block. It is also possible to use state charts to discretize the behavior of breaking curves.

The benefits of SysML for a semi formal modeling of the SUBSET-026-3.5.

- General-purpose modeling languages
- Can model different domain (arithmetic, state charts ...)
- Easy export : As an OMG UML 2.0 profile, SysML models are designed to be exchanged using the XML Meta-data Interchange (XMI) standard.
- Open source licensee for SysML language
- Requirements diagram to capture functional and/or performance requirements.
- Requirements link the SRS to the model
- Easy data structure definition
- Customized language via profile, may be use for domain specific purpose

Enterprise Architect benefits :

- Graphical modeling
- Different view of the system in one tool
- List view of all model elements
- Author, date and status associated to each model element

- Export/Import facilities
- Different view for diagrams (Table views, hierarchy or list view)

Some drawbacks :

- SysML is only semi-formal
- EA is not an open-source software
- EA needs to work with different tools or plug-in to animate and simulate the models that are not always using the same XMI definition.
- The semantic and a glossary should be defined before using SysML
- EA No requirements table representation

5 Detailed Model Description

5.1 Abstraction

The MoRC model has been made by direct translation of the specification described in ERTMS subset-026-3.5 In order to keep the complexity low, some abstractions have been made. The model's behavior would be refined in a next step of the model's design process. In a first step we focus on the communication protocol between the MoRC and the RTM. This leads us to abstract some behaviors.

First, our representation will not consider the interfaces with the JRU since it only recorded existing signals and is not relevant for tests generation Secondly, the orders coming from BTM, DMI, EVC will be abstracted as only one message from the on-board. This message will indicate that a communication session should be started or be ended. We will not distinguished between the different events that may occurred since they follow the same connection protocol. Moreover the discrimination between these events is not well defined in the specification, we will assume that the decision is taken by another task of the EVC and that the MoRC task only starts or terminates a radio communication Finally, the output messages from MoRC to DMI are not considered for the first version.

5.2 Inputs and outputs messages

Figure 6 shows the messages exchanged at the interface of the radio communication management module. The numbers in brackets represent the maximal value the message may have. Note that in a first version the communication will only be set up with an RBC, communication with a RIU (radio in-fill unit) are not taken into account.

5.2.1 RTM interface

`MessageToRBC`, `MessageFromRBC` are the Euroradio messages, their possible values are defined in the subset-026-8.4 and the subset-026-7.4. The test cases define by the subset-076 are performed by analyzing the recording of these messages. In our model we have consider only the relevant messages. Furthermore, messages are decomposed in variable and packets, these are not taken into account by our modeling. Our model considers messages only as a number corresponding to an action. Table 1 summarizes the considered messages, the message name are those used in the model, Id and Packets are those defined in Subset-026-7 and Subset-026-8.

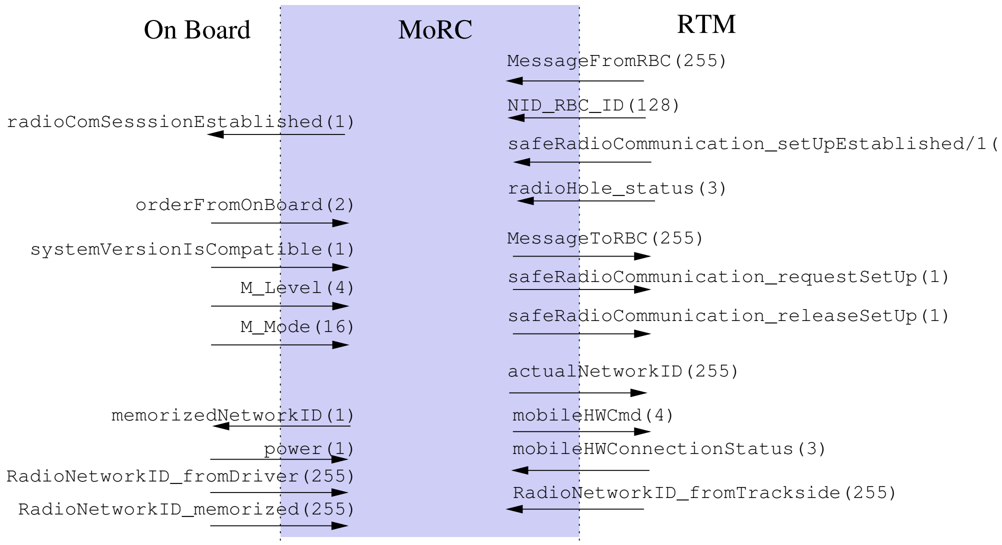

On Board
MoRC
RTM
MessageFromRBC(255)
NID_RBC_ID(128)
radioComSesssionEstablished(1)
safeRadioCommunication_setUpEstablished/1(
radioHole_status(3)
orderFromOnBoard(2)
systemVersionIsCompatible(1)
MessageToRBC(255)
M_Level(4)
safeRadioCommunication_requestSetUp(1)
M_Mode(16)
safeRadioCommunication_releaseSetUp(1)
actualNetworkID(255)
memorizedNetworkID(1)
mobileHWCmd(4)
power(1)
mobileHWConnectionStatus(3)
RadioNetworkID_fromDriver(255)
RadioNetworkID_fromTrackside(255)
RadioNetworkID_memorized(255)

Figure 6. Radio control manager interfaces.

Table 1. Messages exchange during the Management of Radio Communication

Message Name	Id	Packets	Description
TERM_SESSION_TRACK	24	Packet 42 ; Q_RBC = 0	The RBC orders the EVC to **terminate** a communication with RBC
INIT_SESSION_ORDER	24	Packet 42 ; Q_RBC = 1	The RBC **orders the initiation** of a communication
SYS_VERSION	32		The RBC **acknowledge the initiation** of a communication and gives its system version
INIT_SESSION_TRACK	38		The RBC **initiates** a communication
TERM_ACK	39		The RBC **acknowledge** the termination of a communication
NETWORK_REG_ORDER		Packet 45	The RBC orders the **radio network registration**
TRANS_OVER_ORDER	131		The RBC orders a **transition order** over another RBC
SYS_NO_COMP	154		The EVC **acknowledge the establishment with error**.
INIT_SESSION	155		The EVC **initiates** a communication
TERM_SESSION	156		The EVC **terminates** a communication
SESSION_ESTABLISHED	159		The EVC **acknowledge the establishment** of a communication
NO_MESSAGE	255		No messages are send.

NID_RBC_ID identifies the RBC it joins the NID_RBC and NID_C (Subset026-3).

safeRadioCommunication_requestSetUp,safeRadioCommunication_setUpEstablished,safeRadioComm messages are used for setting or releasing a safe radio communication.

radioHole_status may have the value MoRC_rhs_begin, MoRC_rhs_inside, MoRC_rhs_end or MoRC_rhs_none regarding if the train enters, leaves or is in a announced radioHole are compatible.

mobileHWCmd represents the order to register to a radio network given from the OBU to the radio devices.

mobileHWConnectionStatus gives the connection status of the mobile to the radio network.

actualRadioNetworkID is the network ID to register to.

RadioNetworkID_fromTrackside is the network ID given by the trackside

5.2.2 Interface with others on board functions

The different cases to initiate or terminate a communication have been abstract by a single signal. Since the behavior is the same regardless the different events, we assume that others tasks of the EVC will activate the signal wen needed.

orderFromOnBoard represents the order from the on-board EVC to initiate or terminate a communication. The possible values are the following ones :

- MoRC_obo_noOrder: no order;
- MoRC_obo_InitiateCommunication represents one of these cases :
  - Start of mission procedure,
  - Report a mode change,
  - Driver change level to 2 or 3,
  - End of a radio hole,
  - The balise group orders a radio communication.
- MoRC_obo_terminateCommunication represents one of these cases
  - End of mission procedure,
  - Driver closes the desk,
  - Error condition detected on-board.
  - The balise group orders to end up a radio communication.
- MoRC_obo_registeredNetwork orders the radio network registration

systemVersionIsCompatible is set to 1 if a the track and the on-board systems (function system version management)

powerAvailable represents if the power is up or down

memorizeTheLastRadioNetworkID, RadioNetworkID_fromDriver and RadioNetworkID_memorized are used to register the radio devices to the radio network with respects to radio network ID.

The radio management module should take some decision with respect to internal on-board variables. This variable are listed blow. The variable definition are detailed in subset-026-7.

- M_LEVEL ∈ [0..4] represents the levels 0,1,2,3 or NTC.
- M_ Mode ∈ [0..16] represents the on-board operating mode computed by mode function (SUBSET-026.4).

5.3 Internal variables

We define a set of variables use for the radio communication management computation itself.

- countSetUp, countMsg, are used to count the number of try for establishing a radio communication, or to wait for the acknowledgment message (SUBSET-026.3.A).
- msgRecorded: Keep track of MessageFromRBC
- radoComSession ∈ {TERMINATED, ESTABLISHED}: indicates if a radio session is established with the track.
- time used for timeout evaluation

Note that the safeRadio and radoComSession may be used for other OBU functions like the messages given to the driver or mode computation

The constants or the enumeration type such as Modes name we had used in the MorC description are not part of the model itself, they should belong to a special package.

The communication between the MoRC and the radio network registration is done by the following interface.

- safeRadio ∈ {NOCOM, COM, LOST} indicates if a safe radio communication is on.
- mobileSWStatus ∈ {MoRC_mswc_unregistered,MoRC_mswc_registering,MoRC_mswc_registered} indicates the status of the radio network

Note that no communication is possible if no network is registered.

5.4 Behaviour

The behavior is described figure 8. A classical transaction starts with an order to established a communication with an RBC. In this model we assume that the RBC belongs to the RBC accepting list. Note that we have abstracted the different ways to contact an RBC (last known number, number entered by the driver ...). Secondly, the MoRC sets up a safe radio connection, then it initiates a radio communication with the RBC. An order to terminate a radio communication session may occurred, in this case, the MoRC sends a termination message to the RBC waits for the acknowledgment and then releases the safe radio communication. Our model does not manage the consistency of the successive order, we do not impose any constraints to when the orders may occur. This may be done by external tasks.

States INIT_COM and TERM_COM are decomposed as state automaton handling the maximal number of try and the time out of requests.

Sate COM is decomposed as an automaton, it handles the lost of safe radio.

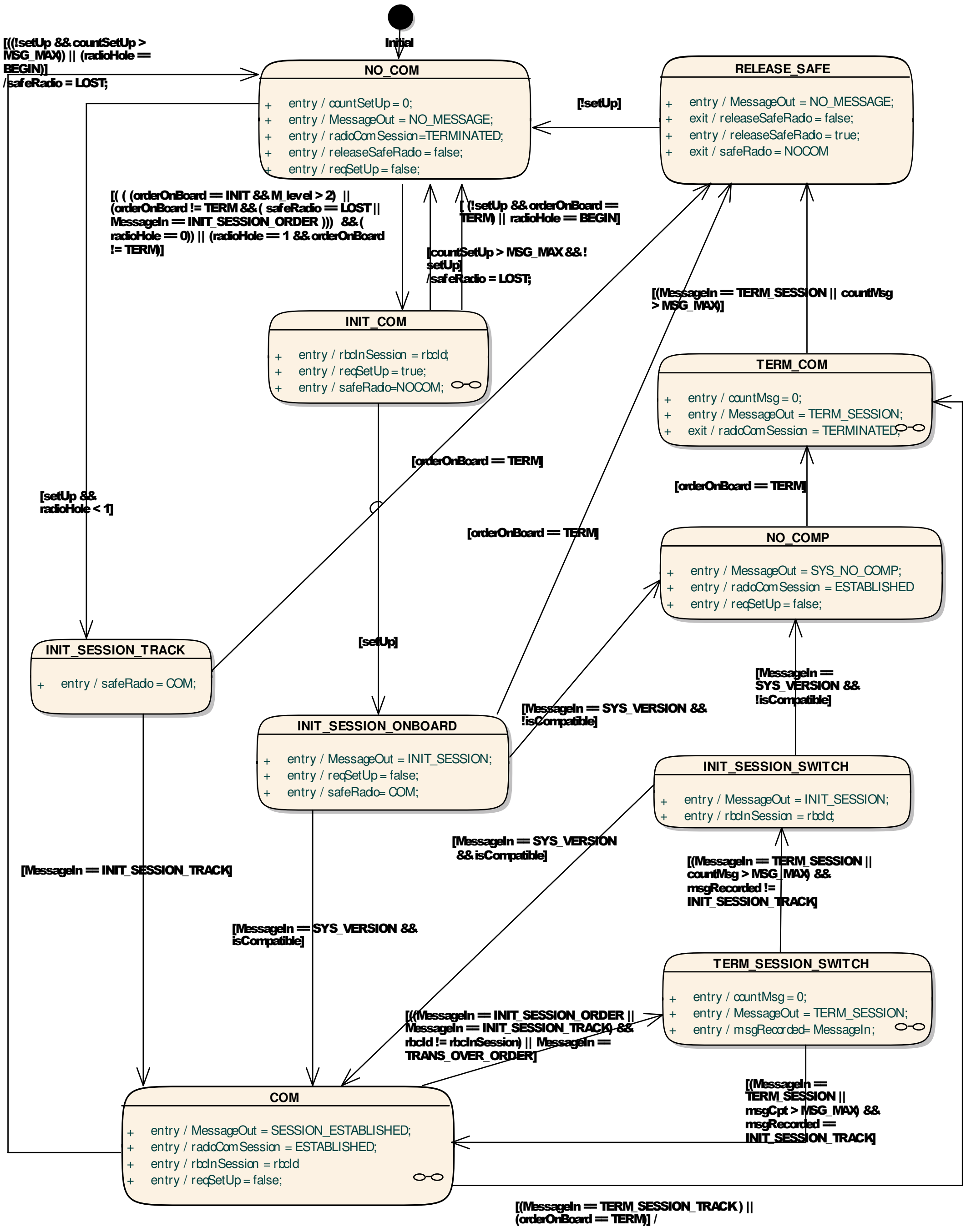

Initial
NO_COM
+ entry / countSetUp = 0;
+ entry / MessageOut = NO_MESSAGE;
+ entry / radioComSession=TERMINATED;
+ entry / releaseSafeRadio = false;
+ entry / reqSetUp = false;
RELEASE_SAFE
+ entry / MessageOut = NO_MESSAGE;
+ exit / releaseSafeRadio = false;
+ entry / releaseSafeRadio = true;
+ exit / safeRadio = NOCOM
[!setUp]
[((!setUp && countSetUp > MSG_MAX)) || (radioHole == BEGIN)]
/safeRadio = LOST;
[(((orderOnBoard == INIT && M_level > 2) || (orderOnBoard != TERM && (safeRadio == LOST || MessageIn == INIT_SESSION_ORDER))) && (radioHole == 0)) || (radioHole == 1 && orderOnBoard != TERM)]
[(!setUp && orderOnBoard == TERM) || radioHole == BEGIN]
[countSetUp > MSG_MAX && ! setUp]
/safeRadio = LOST;
[(MessageIn == TERM_SESSION || countMsg > MSG_MAX)]
INIT_COM
+ entry / rbcInSession = rbcId;
+ entry / reqSetUp = true;
+ entry / safeRadio=NOCOM;
TERM_COM
+ entry / countMsg = 0;
+ entry / MessageOut = TERM_SESSION;
+ exit / radioComSession = TERMINATED;
[orderOnBoard == TERM]
[orderOnBoard == TERM]
[setUp && radioHole < 1]
[orderOnBoard == TERM]
NO_COMP
+ entry / MessageOut = SYS_NO_COMP;
+ entry / radioComSession = ESTABLISHED
+ entry / reqSetUp = false;
INIT_SESSION_TRACK
+ entry / safeRadio = COM;
[setUp]
[MessageIn == SYS_VERSION && !isCompatible]
[MessageIn == SYS_VERSION && !isCompatible]
INIT_SESSION_ONBOARD
+ entry / MessageOut = INIT_SESSION;
+ entry / reqSetUp = false;
+ entry / safeRadio= COM;
INIT_SESSION_SWITCH
+ entry / MessageOut = INIT_SESSION;
+ entry / rbcInSession = rbcId;
[MessageIn == SYS_VERSION && isCompatible]
[MessageIn == INIT_SESSION_TRACK]
[(MessageIn == TERM_SESSION || countMsg > MSG_MAX) && msgRecorded != INIT_SESSION_TRACK]
[MessageIn == SYS_VERSION && isCompatible]
TERM_SESSION_SWITCH
+ entry / countMsg = 0;
+ entry / MessageOut = TERM_SESSION;
+ entry / msgRecorded= MessageIn;
[((MessageIn == INIT_SESSION_ORDER || MessageIn == INIT_SESSION_TRACK) && rbcId != rbcInSession) || MessageIn == TRANS_OVER_ORDER]
[(MessageIn == TERM_SESSION || msgCpt > MSG_MAX) && msgRecorded == INIT_SESSION_TRACK]
COM
+ entry / MessageOut = SESSION_ESTABLISHED;
+ entry / radioComSession = ESTABLISHED;
+ entry / rbcInSession = rbcId
+ entry / reqSetUp = false;
[(MessageIn == TERM_SESSION_TRACK) || (orderOnBoard == TERM)] /

Figure 7. Automaton of the radio communication management

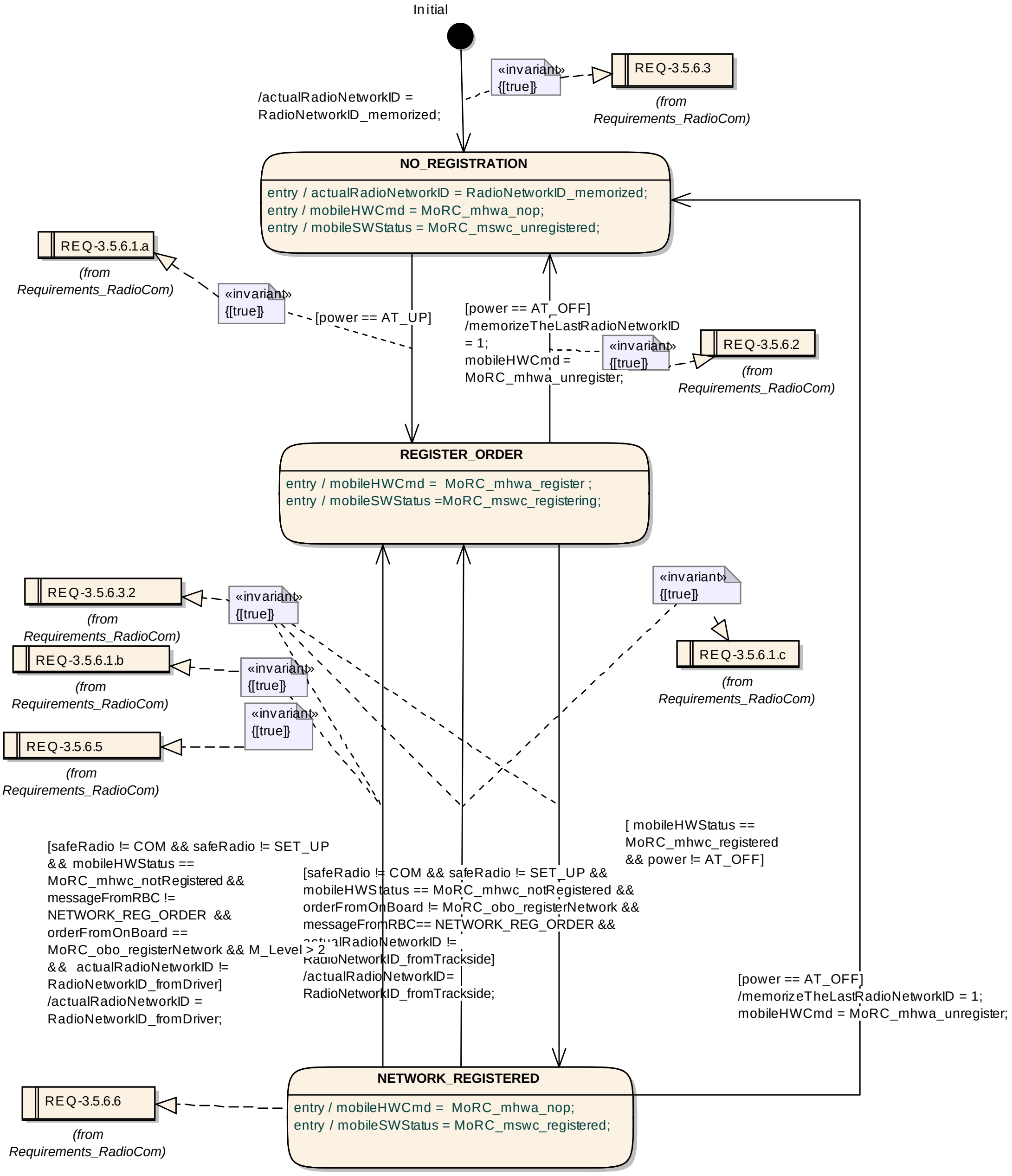

Initial
/actualRadioNetworkID =
RadioNetworkID_memorized;
«invariant»
{[true]}
REQ-3.5.6.3
(from
Requirements_RadioCom)
NO_REGISTRATION
entry / actualRadioNetworkID = RadioNetworkID_memorized;
entry / mobileHWCmd = MoRC_mhwa_nop;
entry / mobileSWStatus = MoRC_mswc_unregistered;
REQ-3.5.6.1.a
(from
Requirements_RadioCom)
«invariant»
{[true]}
[power == AT_UP]
[power == AT_OFF]
/memorizeTheLastRadioNetworkID
= 1;
mobileHWCmd =
MoRC_mhwa_unregister;
«invariant»
{[true]}
REQ-3.5.6.2
(from
Requirements_RadioCom)
REGISTER_ORDER
entry / mobileHWCmd = MoRC_mhwa_register ;
entry / mobileSWStatus =MoRC_mswc_registering;
REQ-3.5.6.3.2
(from
Requirements_RadioCom)
«invariant»
{[true]}
«invariant»
{[true]}
REQ-3.5.6.1.c
(from
Requirements_RadioCom)
REQ-3.5.6.1.b
(from
Requirements_RadioCom)
«invariant»
{[true]}
«invariant»
{[true]}
REQ-3.5.6.5
(from
Requirements_RadioCom)
[mobileHWStatus ==
MoRC_mhwc_registered
&& power != AT_OFF]
[safeRadio != COM && safeRadio != SET_UP
&& mobileHWStatus ==
MoRC_mhwc_notRegistered &&
messageFromRBC !=
NETWORK_REG_ORDER &&
orderFromOnBoard ==
MoRC_obo_registerNetwork && M_Level > 2
&& actualRadioNetworkID !=
RadioNetworkID_fromDriver]
/actualRadioNetworkID =
RadioNetworkID_fromDriver;
[safeRadio != COM && safeRadio != SET_UP &&
mobileHWStatus == MoRC_mhwc_notRegistered &&
orderFromOnBoard != MoRC_obo_registerNetwork &&
messageFromRBC== NETWORK_REG_ORDER &&
actualRadioNetworkID !=
RadioNetworkID_fromTrackside]
/actualRadioNetworkID=
RadioNetworkID_fromTrackside;
[power == AT_OFF]
/memorizeTheLastRadioNetworkID = 1;
mobileHWCmd = MoRC_mhwa_unregister;
NETWORK_REGISTERED
entry / mobileHWCmd = MoRC_mhwa_nop;
entry / mobileSWStatus = MoRC_mswc_registered;
REQ-3.5.6.6
(from
Requirements_RadioCom)

Figure 8. Automaton of the network registration

5.5 Test cases generation

5.5.1 Tool overview

The RT-Tester test automation tool, made by Verified [10], performs automatic test generation, test execution and real-time test evaluation. It supports different testing approach such as unit testing, software integration testing for component, hardware/software integration testing and system integration testing. The RT-Tester version we had used, follows the model-based testing approach [6, 7] and it provides the following features :

- Automated Test Case Generation
- Automated Test Data Generation
- Automated Test Procedure Generation
- Automated Requirement Tracing
- Test Management system

Starting from a test model design with UML/SYML, the RT-tester fully automatically generates test cases. They are then specified as test data (sequences of stimuli with timing constraints) and used to stimulate the SUT and run concurently with the generated test oracles. The test procedure is the combination of the test oracles and the SUT that can be compiled and executed.

The tool supports test cases/data generation for structural testing. It automatically generates reach statement coverage, branch coverage and modified condition/decision coverage (MC/DC) as far as this is possible. The test cases may all be linked to requirements ensuring a complete requirement traceability. Additionally RT-tester may produce test cases/data from a LTL formula, since a LTL formula describes a possible run of the model.

Finally the tool may produce the documentation of tests for certification purposes. For each test cases the following document are produced :

- *Test procedure*: that specifies how one test case can be executed, its associated test data produced and how the SUT reactions are evaluated against the expected results.
- *Test report*: that summarizes all relevant information about the test execution.

In [1], a general approach on how to qualify model-based testing tool according to the standard ISO 26262 ad RTCA DO178C has been proposed and applied with success to the RT-tester tool. Following the same approach compatibility with the CENELEC EN50128 may be easily done.

5.5.2 Test generation

RT-Tester is able to automatically generate test cases from the test model. The first step may be to generate the test cases that cover all the model artifacts : a test to reach all transitions, control state and perform MC/DC coverage of the model. The test cases are related to their attached requirement. This is automatically derived from the requirement diagram.

The test campaign may then be guided by requirement coverage, one can select the test cases associated to a requirement, or by test cases coverage, one select the tests to be covered.

Finally, it is possible to add user define test cases. This has been done to import test cases part from the SUBSET-076. his has been made manually by translating the test cases description from the word document as an LTL formula. For example the test case 1 of feature 177, tests the requirement : "Establishing a radio communication initiated by RBC”. It checks that after receiving a connection indication of the safe radio from the RTM, the OBU shall confirm the establishment and shall consider the session established following the process describes in the requirements 3.5.3. This has been translated into the following formula:

```
// Initial state NO_COM
[
setUp == 0 &&
radioComSesssionEstablished == 0 &&
M_Level >= 3 &&
M_Mode != 3 &&
M_Mode != 4 &&
M_Mode != 5 &&
M_Mode != 9 &&
M_Mode != 10 &&
radioHole == NONE &&
orderOnBoard == NONE  &&
messageIn == NO_MESSAGE
] &&
[
setUp == 0 &&
radioComSesssionEstablished == 0 &&
M_Level >= 3 &&
M_Mode != 3 &&
M_Mode != 4 &&
M_Mode != 5 &&
M_Mode != 9 &&
M_Mode != 10 &&
radioHole == NONE &&
orderOnBoard == NONE  &&
messageIn == NO_MESSAGE
]
Until
(// Safe radio set up
[
  setUp == 1 &&
  IMR.messageFromRBC == 38
] &&
Finally
// Message session established send
// session considered as established
[
  IMR.messageFromRBC == 38 &&
  IMR.messageToRBC == 159 &&
  IMR.radioComSesssionEstablished == 1
])
```

Test cases covered	# tests generated
Basic state	14
Transition	34
MC/DC	85
LTL	4
Test 177	2
Total Tests	139

Total Requirements cover 40

Table 2. Test cases generation summary

Table 2 summarizes the set of test cases generated and the related requirements. The line LTL represents the requirement that has been rewrite with a LTL formula. The test 177 represents the translation as LTL formula of the two test cases described in SUBSET 076. This set of test cases covers 36 requirements from SUBSET-026-chap 3.5.

References

[1] Jörg Brauer, Jan Peleska, and Uwe Schulze. Efficient and trustworthy tool qualification for model-based testing tools. In Brian Nielsen and Carsten Weise, editors, *Testing Software and Systems*, volume 7641 of *Lecture Notes in Computer Science*, pages 8–23. Springer Berlin Heidelberg, 2012.

[2] Eclipse Project Papyrus. Papyrus. http://www.eclipse.org/papyrus/, 2011.

[3] David Mentre, Stanislas Pinte, and Guillaume Pottier. D2.5 Methods and tools benchmarking methodology. Technical Report March, ITEA2 openETCS consortium, 2012.

[4] Object Management Group. OMG Systems Modeling Language (OMG SysML^{TM}). http://www.omgsysml.org, June 2012.

[5] Object Management Group (OMG). OMG Object Constraint Language (OCL), 2012.

[6] Jan Peleska, Elena Vorobev, and Florian Lapschies. Automated test case generation with smt-solving and abstract interpretation. In Mihaela Bobaru, Klaus Havelund, GerardJ. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 298–312. Springer Berlin Heidelberg, 2011.

[7] Jan Peleska, Elena Vorobev, Florian Lapschies, and Cornelia Zahlten. Automated model-based testing with RT-Tester. Technical report, Universität Bremen, 2011.

[8] Amir Pnueli. The temporal logic of programs. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 46–57, 31 1977-Nov. 2.

[9] Sparx System. Enterprise Architecture. http://www.sparxsystems.com/products/mdg_sysml.html, 2009.

[10] Verified Systems International GmbH. Verified :: Products. http://www.verified.de/en/products.