OETCS

openETCS

ITEA2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

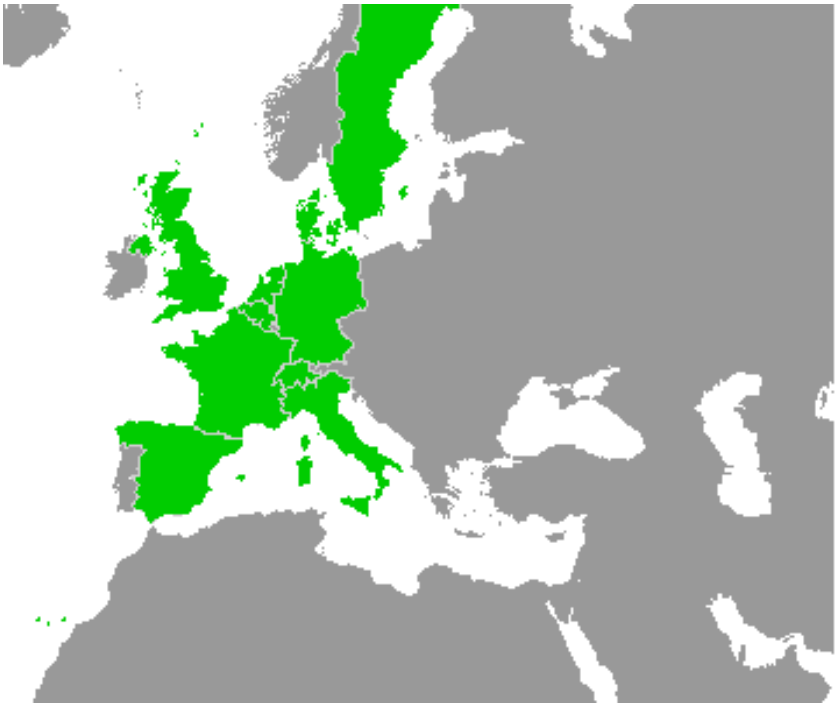
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Toolchain”

Radio Communication Managment
SRS SUBSET-026-3.5
SysML Model

Cécile Braunstein

March 2013



This page is intentionally left blank

Work-Package 7: “Toolchain”

OETCS
March 2013

Radio Communication Managment
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.

Table of Contents

1 Short Introduction to Formalism and Tool 5
2 Modeling Strategy 5
3 Model Overview 5
3.1 Block and statechart view 5
3.2 Requirement view 7
4 Model Benefits 8
5 Detailed Model Description 9
5.1 Abstraction 9
5.2 Inputs and outputs messages 9
5.3 Internal variables 11
5.4 Behaviour 12
References 12

Figures and Tables

Figures

Figure 1. Interactions of the MoRC and others On-board modules 6
Figure 2. Model Overview 6
Figure 3. Example of modeling a requirement 7
Figure 4. Radio control manager interfaces. 9
Figure 5. The MoRC class 12
Figure 6. Automata of the radio communication management 13

Tables

Table 1. Messages exchange during The Managment of Radio Communication 10

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 [2].

The formalism used is SysML [3]. More particulary we had used block diagram, statecharts and requirements diagram. SysML is a graphical language that extends UML for a custumized version suitable for system engieneering. It may help modeling system whithin a board range of system variety that may include hardware, software, data, personel and facilities. It supports the specification, analysis design, verification and validation of complex system.

Enterprise architect (EA) version 9.3 [6] has been used to implement the model. Note that this tool is not open source but others tools such as Papyrus [1] provides SysML modeling capabilities and are evaluate by others parteners. 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 devellopment cycle. The main advantages is the requirement managment 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) [4, 5]. It provides sequences of input data with timing constraints that are used for the stimulation of the sysem under test (SUT), concurently 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 subpart of the specification define 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 statechart diagram. In parralel, we implemented a requriement diagram that make the link between the requirement list and the SRS to the model.

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 the determination 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).

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 and the TE model and a set of constants and enumeration types. The requirement package contains all the requirement and a requirement diagram. package.

3.1 Block and statechart view

The radio communication management module (MoRC) is the user application interacting with Euroradio protocol. In this section we present the general behaviour only, the actual details of how messages are transported are not relevant for this description.

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 1:

- (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 (cf. Subset-026-4.5) used by the MoRC such as:
  - track conditions managment functions (Subset-026-3.12.1) that determine for instance if the ECTS system of the track is compatible with the system on-board.
  - debug purpose.

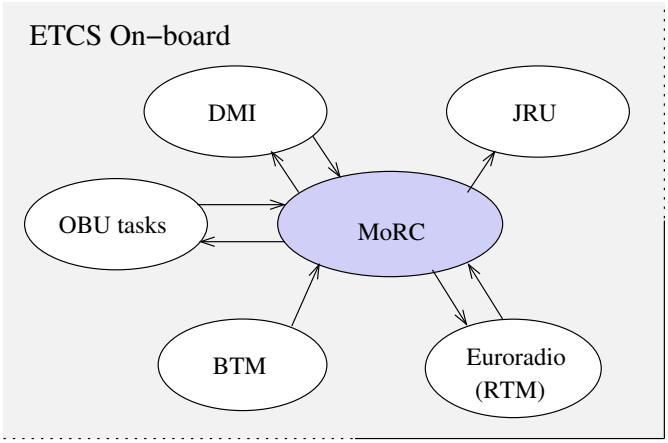


Figure 1. Interactions of the MoRC and others On-board modules

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 section5.1 for more details).

In figure 2 the MoRC model - *our test model* - is composed with a the test environment (TE). The inputs and outputs interfaces are explained in detail in section 5.2. The test environment abstracts away all the others functions or blocks which the SUT may interract with.

The SUT block consits here of unique block that is the MoRC. The MoRC block contains the internal variables used for the protocol implementation and the statecharts describing its behavior. For the moment the constants used in this chapter and defined in SUBSET-026.3-A.3.1 are not part of the MoRC block. At this stage of modelisation it is not defined where they shuld 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.

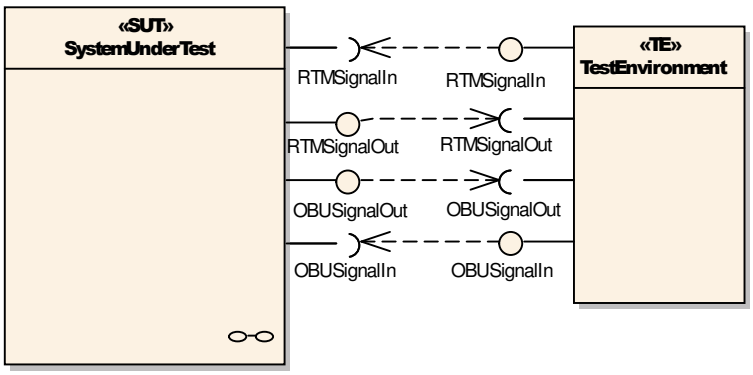


Figure 2. Model Overview

3.2 Requirement view

SysML offers a modeling constructs to represent text-based requirements and relationship with other odeling elements. The requirements in EA diagram may be described in a graphical or tree strucured 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 xml 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 automaticaly 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 3 shows an example of the requirement representation.

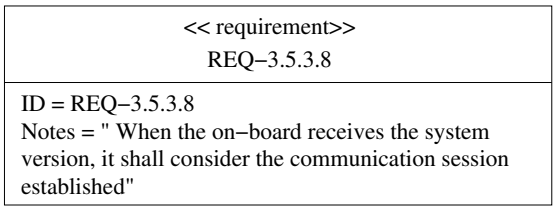


Figure 3. Example of modeling a requirement

Most of requirement refers to the bahaviours model as transition of the statechart 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 behavoir 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. SysML does not imposed

any language, one can choose other constraints language such as OCL. The following example shows the translation of two requirements into a LTL properties. The reader should refer to sectio 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 ==25 || MessageIn == 38  )
&& Finally (MessageOut == 159))
```

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 == 156 && setUp == 1 ->
X (MessageOut == 159 && radioComSession == 1)
```

The requirement diagram only represent the "satisfy" relations between the requirements and the model. One could refine this diagram and add derive dependency or containement relationship. Note that in practice we could show the "satisfy" relations directly on the statechart view. The seperate 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 arthmetic may be represented as parametric diagram with constraints block. It is also possible to use state charts to discretise the behavior of breaking curves.

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

- Graphical modeling
- 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 Metadata Interchange (XMI) standard.
- Easy import, for requirement and model part
- Open source licence for SysML language
- Requirements diagram to capture functional, performance requirements.
- Requirements link the SRS to the model
- Table views
- Easy data structure definition

EA

- 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

Some drawbacks

- Only semi-formal
- EA is not an open-source software
- EA needs to work with different tools or plugin for animate and simulates the models that are not always using the same xmi definition

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 behaviour 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 behaviours.

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 occured 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 Finaly, the output messages from MoRC to DMI are not considered for the first version.

5.2 Inputs and outputs messages

Figure 4 shows the messages exchanged at the interface of the radio communication managment 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

`MessageOut`, `MessageIn` 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 analysing 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.

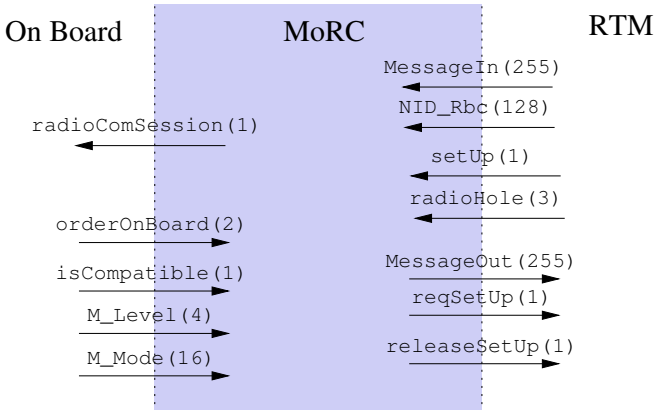


Figure 4. Radio control manager interfaces.

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 **acknoledge** the termination of a communication
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.

Table 1. Messages exchange during The Managment of Radio Communication

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

`setUp`,`reqSetUp`,`releaseSetUp` messages are used for setting or releasing a safe radio communication.

`radioHole` may have the value `BEGIN`, `INSIDE`, `END` or `NONE` regarding if the train enters, leaves or is in a announced radioHole are compatible.

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.

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

- `NONE`: no order;
- `INIT` 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.
- `TERM` 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.

`isCompatible` is set to 1 if a the track and the on-board systems (function system version managment)

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` $\in$ [0..4] represents the levels 0,1,2,3 or NTC.
- `M_ mode` $\in$ [0..15] 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. Figure 5 shows the block element.

- `countSetUp`, `countMsg`, are used to count the number of try for establishing a radio communication, or to wait for the acknoledgment message (SUBSET-026.3.A).

- `msgRecorded`: Keep track of `MessageIn`
- `safeRadio` ∈ {`NOCOM`, `COM`, `LOST`} indicates if a safe radio communication is on.
- `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.

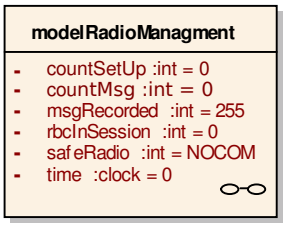


Figure 5. The MoRC class

5.4 Behaviour

The behaviour is described figure 6. 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 acknowledgement 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 automata 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.

References

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

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

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

[4] Jan Peleska, Elena Vorobev, and Florian Lapschies. Automated test case generation with SMT-solving and abstract interpretation. Number 1, pages 298–312, 2011.

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

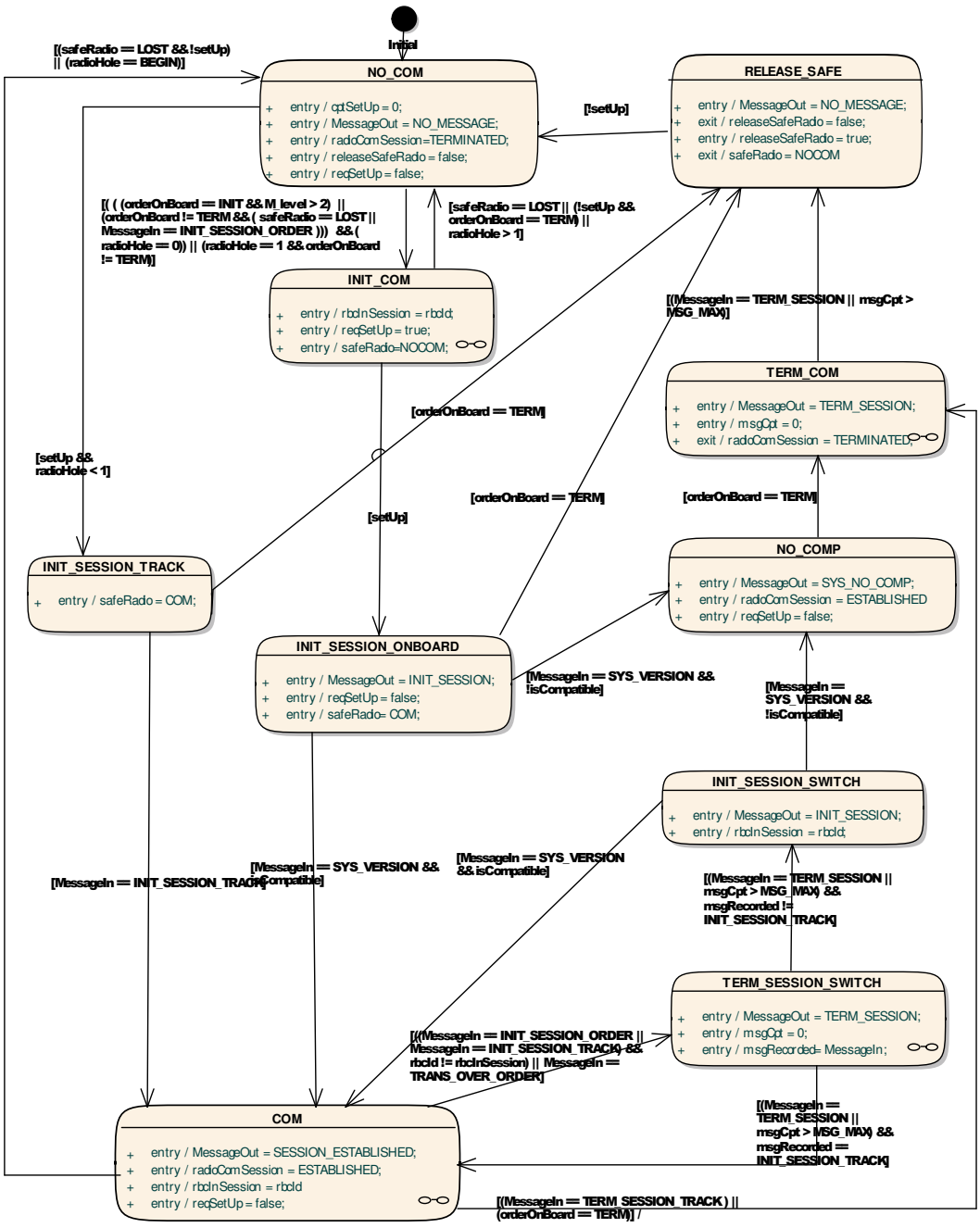


Figure 6. Automata of the radio communication management

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

