OETCS/WP7/O??

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

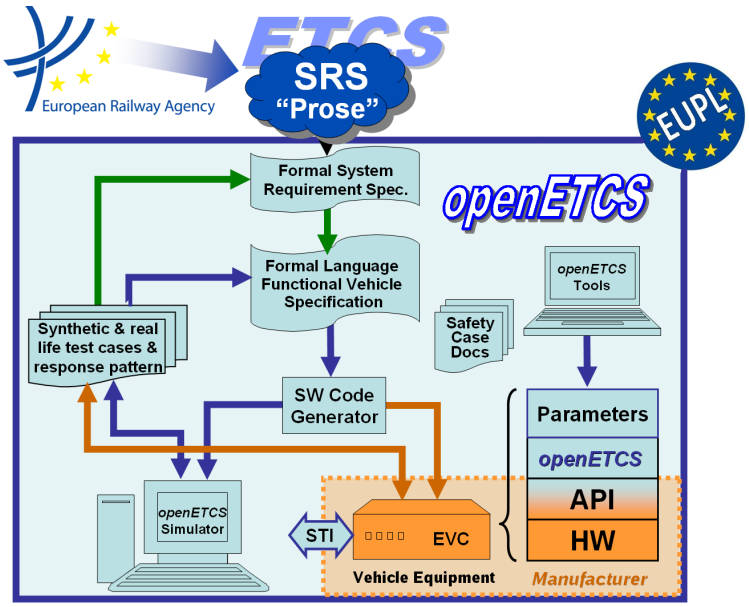
openETCS@ITEA Work Package 7: “Toolchain”

Evaluation model of ETCS using SysML and Enterprise Architect

Tool and model presentation

Thomas Bardot

March 2013



Funded by:

Federal Ministry of Education and Research
Région de Bruxelles-Capitale
GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

This page is intentionally left blank

openETCS@ITEA Work Package 7: “Toolchain” **OETCS/WP7/O??**
March 2013

Evaluation model of ETCS using SysML and Enterprise Architect

Tool and model presentation

Thomas Bardot

Mitsubishi Electric R&D Centre Europe
1 allée de Beaulieu
CS 10806
35708 RENNES cedex 7

email: t.bardot@fr.merce.mee.com

Draft Report

Prepared for openETCS@ITEA2 Project

Abstract: FIXME

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

This document describes a SysML model of the ETCS SRS SUBSET-026-3.5.3 [2]. An overview of the formalism and of the SysML model is given. Then, we discuss about the benefits and shortcomings of the formalism and the modeling strategy. In order to make the model easier to understand, a detailed description of the model with explanations is given.

1.1 Formalism and tool description

1.1.1 SysML

The System Modeling Language (SysML) is a graphical language based on an Unified Modeling Language (UML) profile. It represents a subset of UML 2.0 with extensions needed to model complex systems with hardware, software, procedures and many other elements. SysML uses nine kind of diagrams which allow to model requirements, parameters relationship, structure and behavior of a system. These diagrams are the following:

- Activity diagram;
- Block definition diagram;
- Internal block diagram;
- Package diagram;
- Parametric diagram;
- Requirement diagram;
- Sequence diagram;
- State machine diagram;
- Use case diagram.

SysML diagrams are structured as shown in figure 1, page 2.

1.1.2 SysML tools

There are many tools supporting SysML, open source or not. Most common open source tools are Topcased and Papyrus, based on the Eclipse Platform.

We tried Topcased, but we faced problems, including a corrupted project file. We also had problem when changing the diagram hierarchy : when trying to allocate an existing diagram (bdd) into another object by using the outline view, nothing happened, the *.xmldi* file seems to not be properly updated. We noticed an important memory consumption (up to 1GB), probably due to the Eclipse Platform, that could cause the crash of the tool.

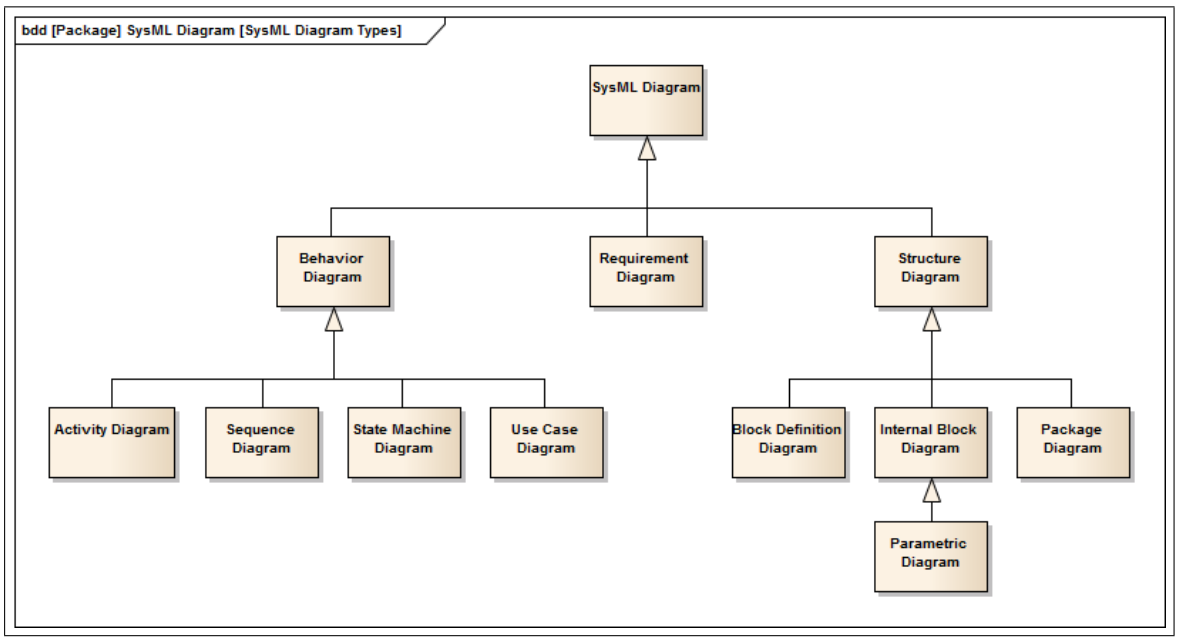


Figure 1. SysML diagram types

Thus, we chose to use Enterprise Architect v10, a tool under a proprietary license edited by Sparx System. It has the advantages to be one of the cheapest proprietary SysML tool. Among its characteristics, Enterprise Architect has a model simulator that permits to simulate behavioral diagram step by step. It also has a code generator that permits to translate diagrams and SysML elements into code in ADA, C/C++, java and other languages. These characteristics could be interesting for further studies about the development of a model simulator. Sparx System claims that Enterprise Architect has a good scalability. Our model is too light to confirm this assertion, but we notice a low memory consumption compared with Topcased, and the tool remains reactive.

1.2 Model purpose

Our work consisted in modeling the SRS SUBSET-026-3.5.3 [2]: “Establishing a communication session”, only when the communication is initiated by the OBU. The purposes of this work is to realize a behavioral model of this ETCS function. The model shall contain all the information states in the [TODO add ref] SRS SUBSET-026-3.5.3, and it shall be easy to access to this information by reading the model. An important point is that the model shall be well structured in order to help its comprehension and verification. The model should also include traceability information.

2 Model overview

2.1 Diagram choice

As seen in chapter 1.1.1, behavioral model can be based on state machine diagram, activity diagram, sequence diagram or use case diagram. A model evaluation with SysML state machine diagrams is already done by Cécile Braunstein [1], so we prefer to evaluate other way to model system behavior. Use case diagram has a too high-level view of the system in order to model detailed functional behavior. Sequence diagram may fit better to the modeling of the SRS SUBSET-026-3.5.3, but it is a particular approach, service-oriented, which may not be relevant to model other ETCS functions.

Therefore, our behavioral model is based mostly on the activity diagram. Activity diagram may describe a behavior by the transformation of inputs to outputs through a controlled sequence of actions. An activity can also depict the behavior performed by a block or a part. Thus, this kind of diagram permits to show the relationship between block inputs and outputs, and to clearly identify the processing performed by a block or a part.

We also used requirement diagram and block definition diagrams in order to collect requirements and to structure our model. Notice that if we had to model more precisely the logical structure of the ETCS functions, we would also use the internal block diagram.

2.2 Modeling strategy

The modeling strategy is represented by the figure 2, page 4.

First of all, the modeling activity started with the definition of all the requirements applicable to the function *Establishing Communication Session*. These requirements simply represent the SRS SUBSET-026-3.5.3 [2] which is split into basics requirements.

Then a brief logical structure of the *Manage Communication Session* functions is made by using blocks and parts. An activity is associated to the block *Establishing Communication Session* and represents its behavior. This activity call other activities which permits to break down the model complexity and to help its comprehension.

In conjunction with the activity modeling, data types are defined by using enumeration and signals.

2.3 Modeling structure

The model is composed of three main packages:

- *Data* package which contains block definition diagrams where enumeration and signals are defined;
- *System requirements* package;
- *System behavior and structure* package.

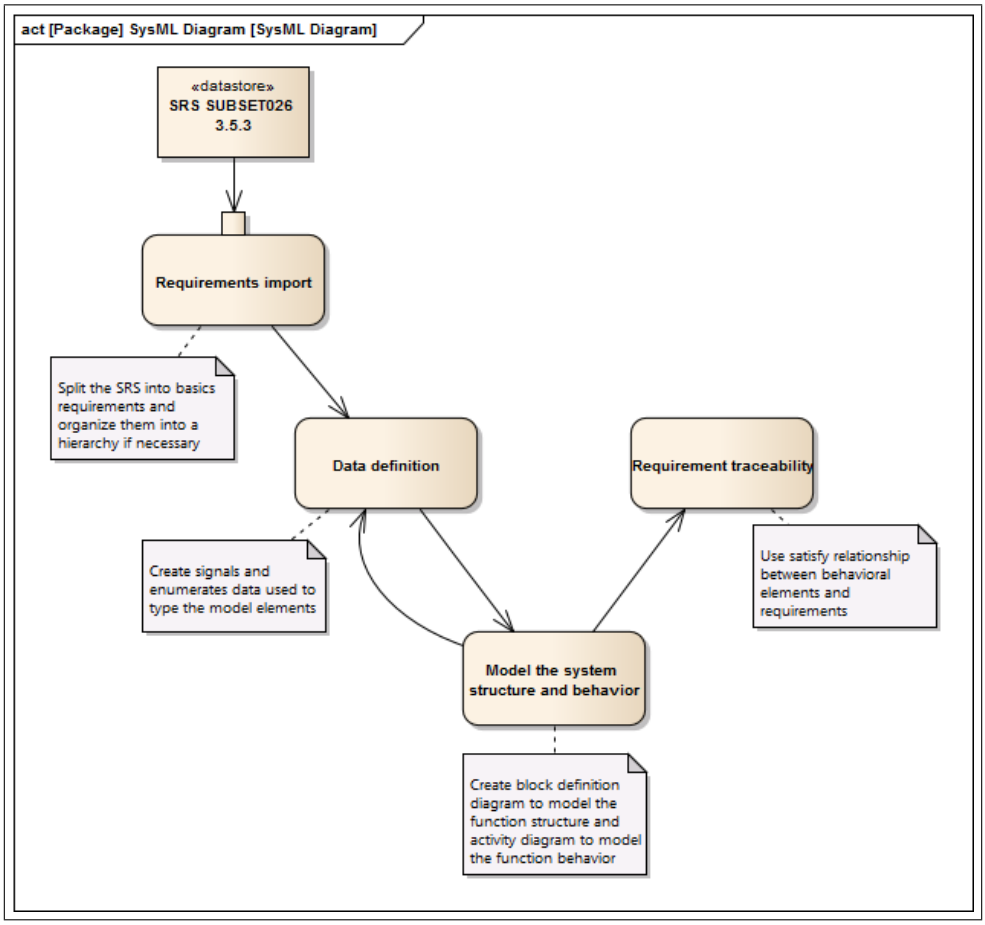


Figure 2. Schematic representing the methodology used

The *system behavior and structure* package imports the *System requirements* and *data* packages as shown by the figure 3, page 5.

2.3.1 *Requirements* package

As explains in chapter 2.2, this package contains requirement diagram where all the requirements applicable to the *Establish a communication session* function are shown. All these requirements simply represent the text of the SRS SUBSET-026-3.5.3 [2], when each requirement represents an elementary part of it.

Because creating manually all these requirements is too long, we generate them by importing a *.csv* file with the format described in the table 1. The *Type* field is a mandatory field that must match Enterprise Architect element types : requirement or package in our use case. In order to organize the model, it is important to import a package and indicate where the imported requirements have to be placed. The model hierarchy is then created with the *CSV_KEY* and *CSV_PARENT_KEY* fields. A key consists in a unique identifier for each element.

Because each requirement represents an elementary part of the SRS text, we try as far as possible to not insert hierarchical dependencies between requirements. Nevertheless, we did it when the SRS specifies a list of conditions that shall be met in order to fulfill a need. In this case, all the requirements corresponding to the conditions are contained into a parent requirement which represents the need to fulfill. The table 1 show an example of this case. Requirement name is made on this pattern : REQ_SRS026_*chapter*.

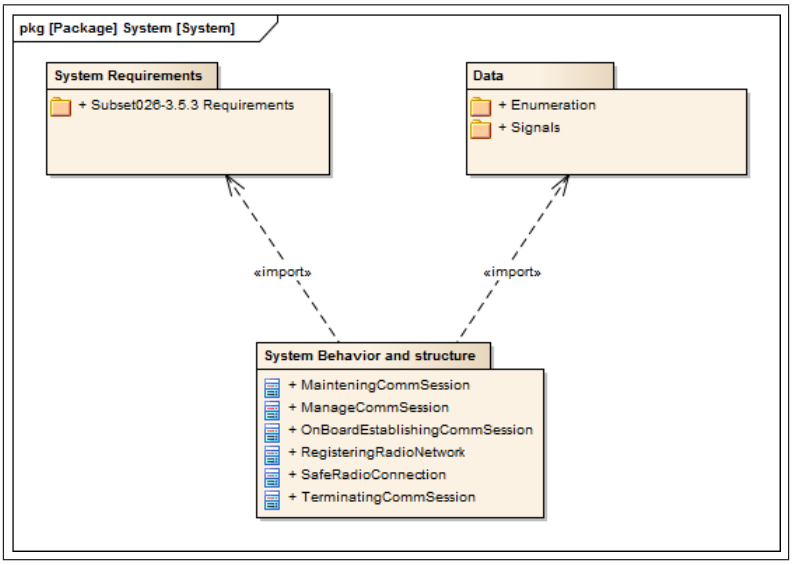


Figure 3. Overview of the SysML model structure

NAME *Name of the element to be imported*	**TYPE** *type of the element (package or requirement)*	**NOTE** *Description of the element*	**CSV_KEY** *Id of the element, same as the element's name*	**CSV_PARENT_KEY** *Id of the hierarchical parent*
Subset026-3.5.3 Requirements	package	Establishing a communication session	Subset026-3.5.3 Requirements	
REQ_SRS026_3.5.3.4	Requirement	The on-board shall establish a communication session	REQ_SRS026_3.5.3.4	Subset026-3.5.3 Requirements
REQ_SRS026_3.5.3.4-1	Requirement	a) At Start of Mission (only if level 2 or 3).	REQ_SRS026_3.5.3.4-1	REQ_SRS026_3.5.3.4

Table 1. format of the .csv file used to import requirements

2.3.2 *Data* package

The *data* package contains block definition diagrams where enumerations and signals are defined (see chapter 4.2). These data types are used in the model to type the different SysML elements like connectors, ports, action pins, activity parameters, attributes or operation arguments.

We choose to use signals to model structured data exchanged between blocks or activities. Indeed, SysML definition of signal is [3]: "A signal defines a message that can be sent and received by a block. It has a set of attributes that specify the content of the message". Then, in our model, a signal element represent a message, and signal attributes represents the message contents. An attribute can be typed with a primitive value (e.g. boolean, integer) or with an enumerate.

2.3.3 *System Behavior and structure* package

The *system behavior and structure* package contains the block definition diagram which describes the functions logical structure (see figure 9, page 10). The physical ETCS structure, for example with an on-board block and a trackside block, is not represented. Thus, blocks only represent ETCS functions or sub-functions which apply to any physical part of the ETCS system.

In our model, we create a block *OnBoardEstablishingCommSession*. It represents the function *Establishing Communication Session* described in the SRS SUBSET-026-3.5.3, only in the case where OBU is initiating the communication session. But, the behavior on this function cannot be

allocated to a particular physical sub-system like the OBU. In fact, this function describes the behavior of different ETCS sub-systems (OBU, RIU or RBC) when the OBU is initiating the communication session.

A block may have attributes or operations. Block attributes should represent information relevant to the block behavior that has to be stored. It can be block states, or constants for example. A block operation defines in our model a basic logical function used to describe a small part of the block behavior. It permits to reduce the complexity and to increase the readability of the behavior diagrams by calling operations. But, on the other hand, diagrams contain less information.

System behavior and structure package also contains the activity diagrams which depict the function behavior. System behavior and system logical structure are in the same package because it helps to link activity to their owner block and improves the model hierarchy.

An activity is associated to the block *OnBoardEstablishingCommSession* and depict its behavior (see figure 10, page 11). This activity has the same name than this block. The inputs and outputs parameters of this activity represents the inputs and outputs of its owner block.

Even if it is not mandatory, our activity diagrams consists in a sequence of actions related by control relationship and this sequence is delimited by an initial node and a final node. It is justified because each activity that we use represents a specific process limited in time with states and not just a never-ending process on a stream.

An activity can call other sub activities. In order to do that, we chose to use call behavior actions. These actions may have some input or output pins corresponding to the input and output parameters of the activity which they call. The connection between activity parameters and action pins is often hidden for a better diagram readability. But we notice that the tool does not provide layers in order to easily hide or show the connectors or another specific part of a diagram.

As mentioned before, an activity can call operations defined in a block. It is done by using a call operation action. The operation input parameters are linked to the action input pins. The operation result is linked to the action output pin.

Activities use decision element that indicates a point of conditional progression: if a condition is true, then processing continues one way; if not, then another. The condition is made by evaluating a variable. This variable often consists in an output pin of a call operation action. So, we assume that an action pin, or an activity parameter, or a block port can be seen as a variable with the same name. We also assume that a connection between two of these elements can be seen as an assignment of the target element with the value of the source element.

3 Model benefits and shortcomings

Using SysML with Enterprise Architect in order to model a SRS chapter presents some benefits and shortcomings which are listed below.

3.1 Benefits

From our point of view, using SysML provides these benefits:

- It is a graphical language easy to understand even to people who are not intimate with this method;
- SysML allows to represent different views of a system, for example a requirements view with requirements diagrams, a view of the system structure with block definition diagram, a view of the system behavior with states machines or activity diagrams, and a view of the system constraints with the parametric diagrams.

From our point of view, Enterprise Architect SysML tool provides these benefits:

- Code generation and behavior simulation;
- Many examples and a useful learning center are provided with the tool;
- The tool seems stable and scalable.

3.2 Shortcomings

SysML doesn't have a strong semantic, therefor when modeling a system, the developer faces several modeling choices. For example, when modeling the data exchange between activities, we had the choice between using send signal action or pass through outputs and inputs activities parameters. These choices have their own advantages and defects. So it is important to set a subset of SysML applicable to the OpenETCS project and to define our semantic.

As mentioned previously, because SysML is a graphical language, it is easy to understand. But it implies that all the model information cannot be on the model diagram. Indeed, if diagram contains to much graphical information, it would become not understandable at all. The requirements management is a good example of this point of view. Requirement diagrams are useful to show dependencies between requirements, but it becomes useless when a lot of requirements are imported to the model. Moreover, using the *satisfy* relationship by drawing a connector between all the model elements and the satisfied requirements can make the diagram not readable, thus the use of a cross-check matrix could be more relevant.

Enterprise Architect allows model code generation or model simulation. But we need to create a particular model in order to be able to generate code or simulate it. The model simulation permits to simulate states machines or activities one by one. But we did not find how to simulate several activities diagrams running concurrently.

4 Detailed model description

This chapter gives all the diagrams used by our model. But, because all the model information can't stand only on diagrams, please refer to the automatically generated *ModelDocumentation.pdf* for a full model description.

4.1 *Requirements* package

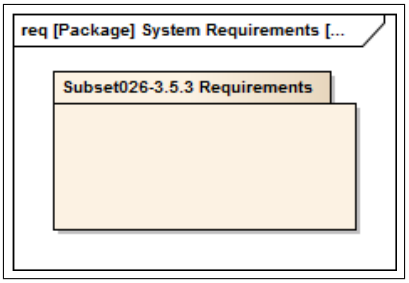


Figure 4. Req diagram containing packages for each ETCS chapter

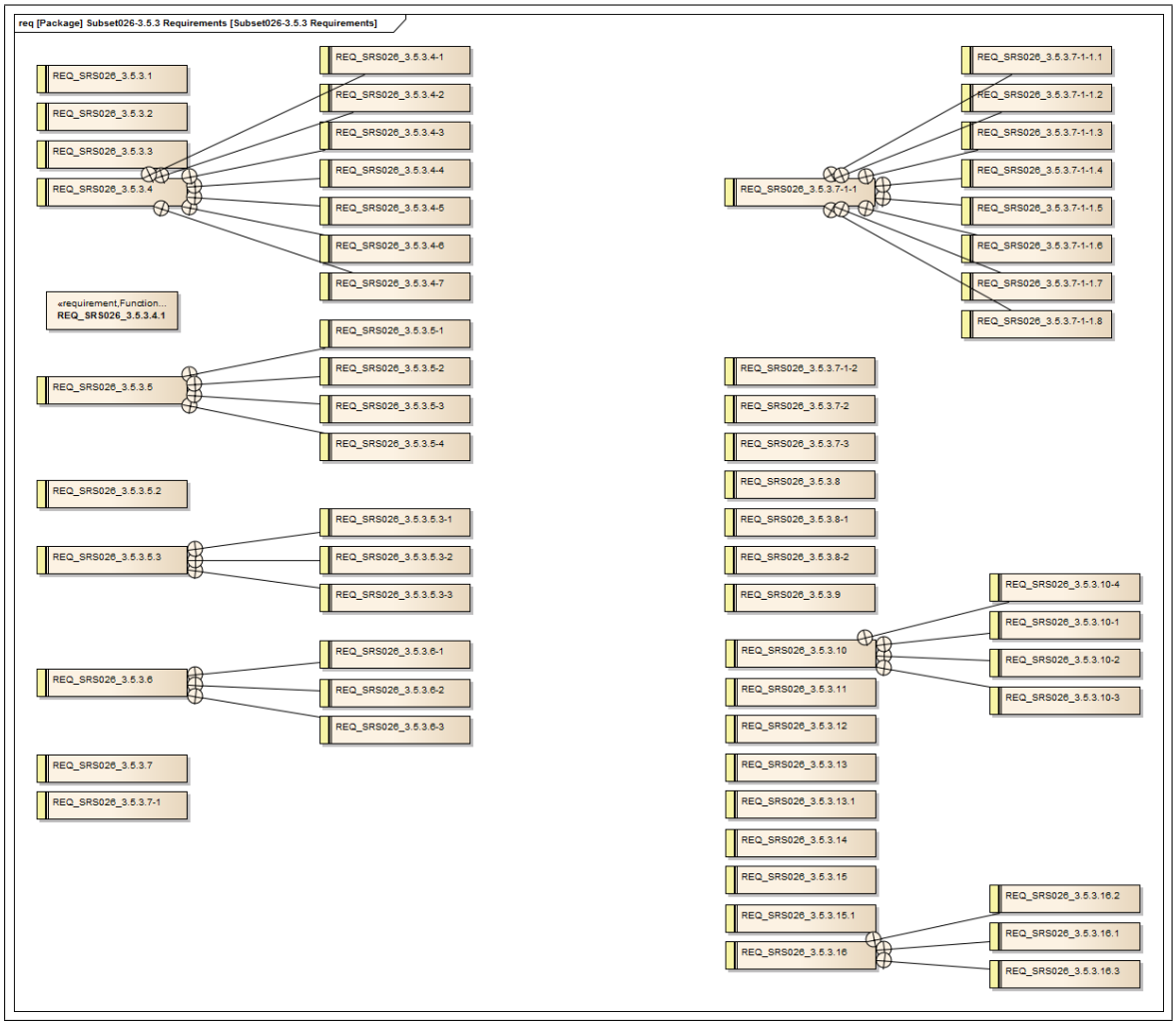


Figure 5. Req diagram containing the requirements corresponding to the SRSSYBSET026 3.5.3

4.2 *Data* package

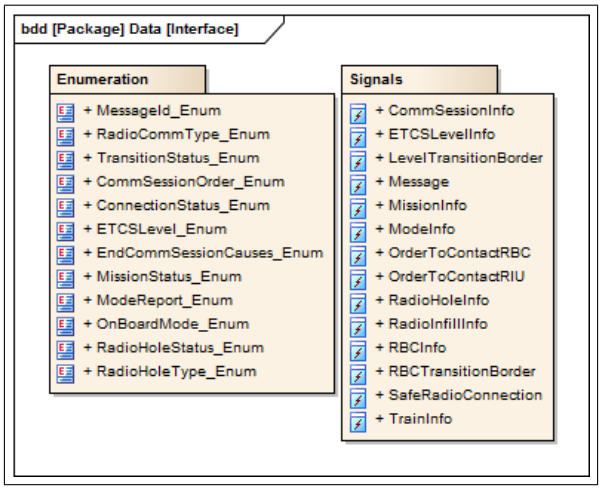


Figure 6. Bdd diagram containing packages for each data type (enumeration or signal)

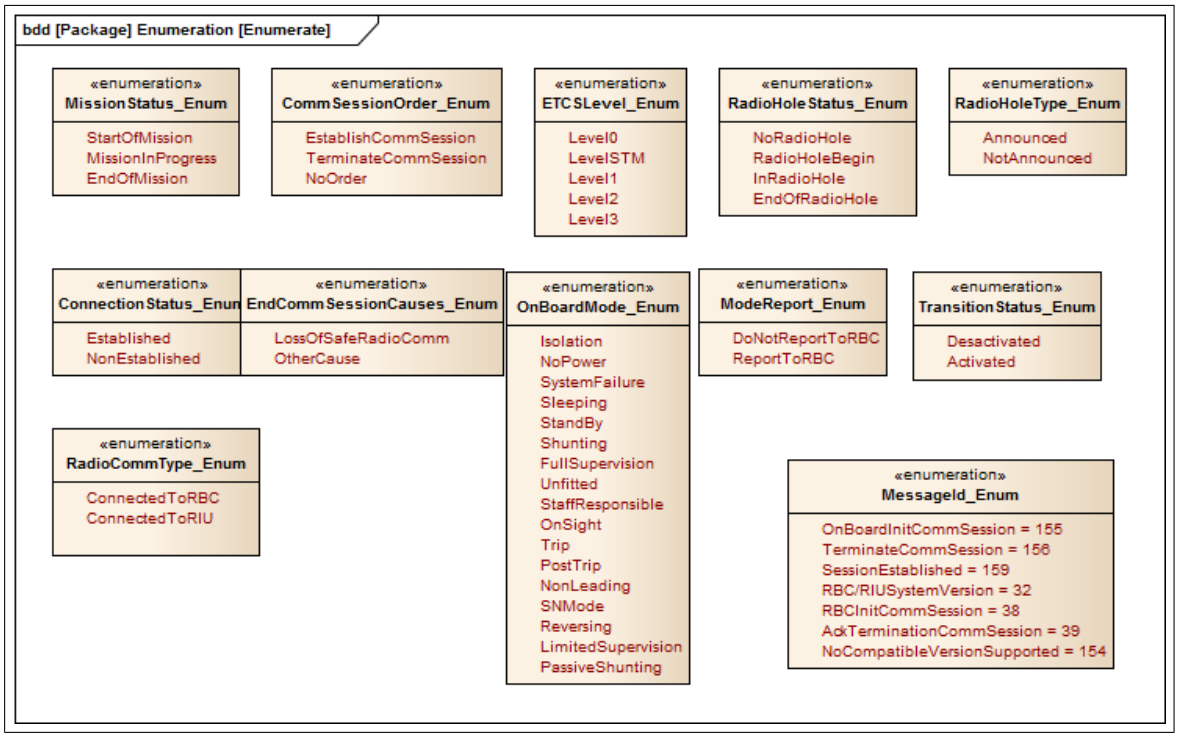


Figure 7. Bdd diagram containing the definition of the enumerates used by the model

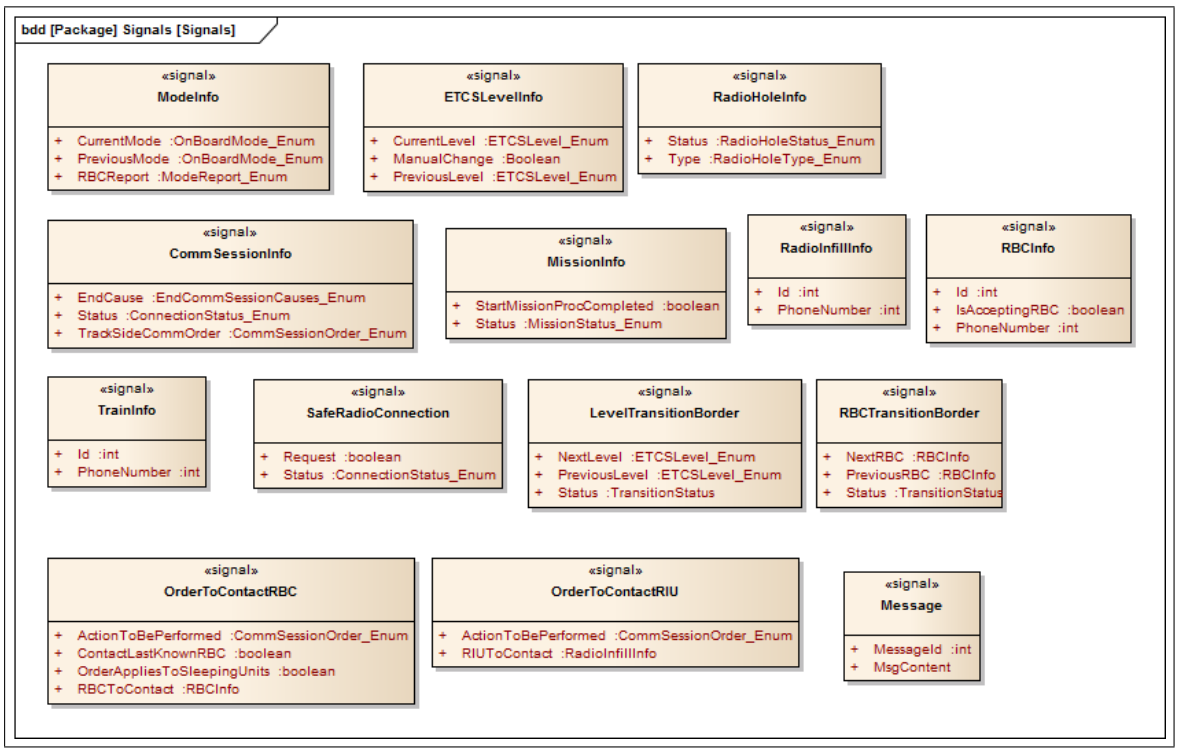


Figure 8. Bdd diagram containing the definition of the signals used by the model

4.3 *System Behavior and structure* package

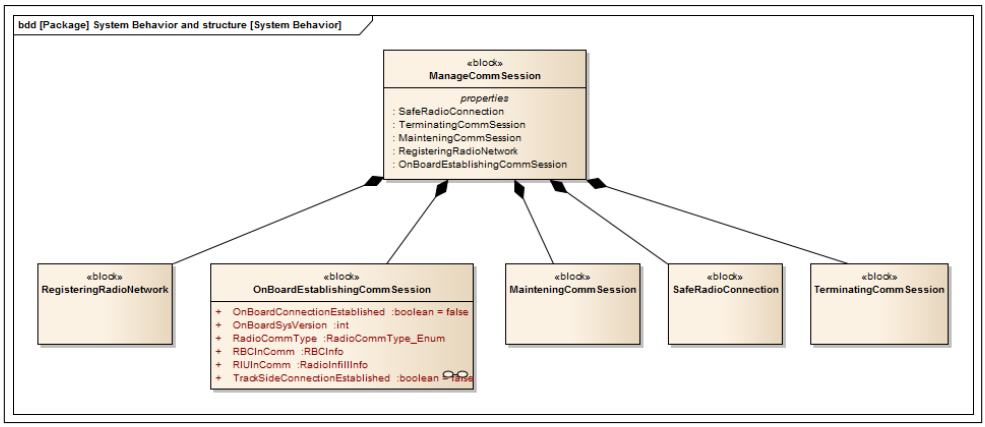


Figure 9. Bdd diagram containing blocks which provide a very brief description of the function structure

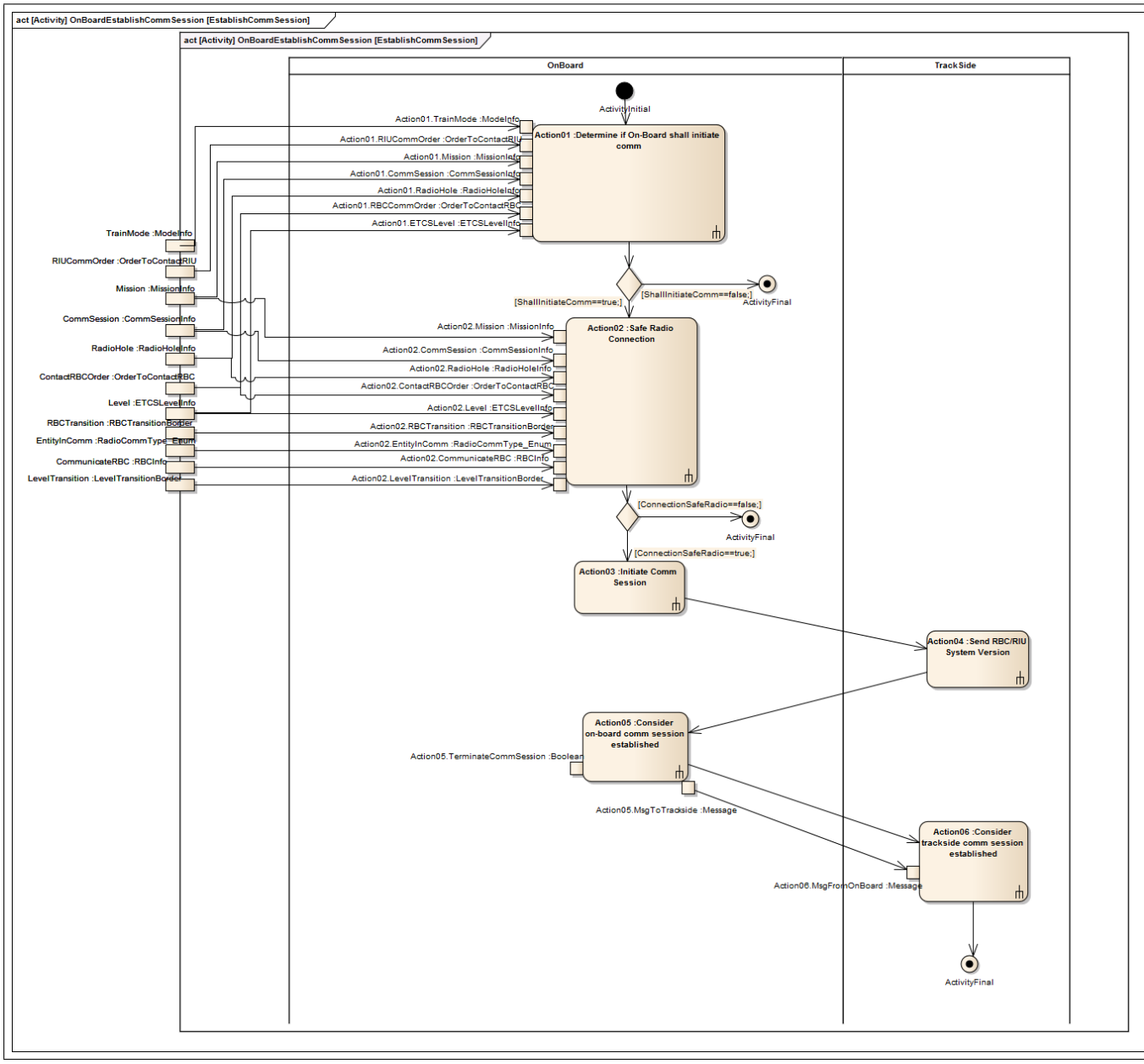


Figure 10. Main activity diagram describing the behavior of the block *OnBoardEstablishingCommSession*

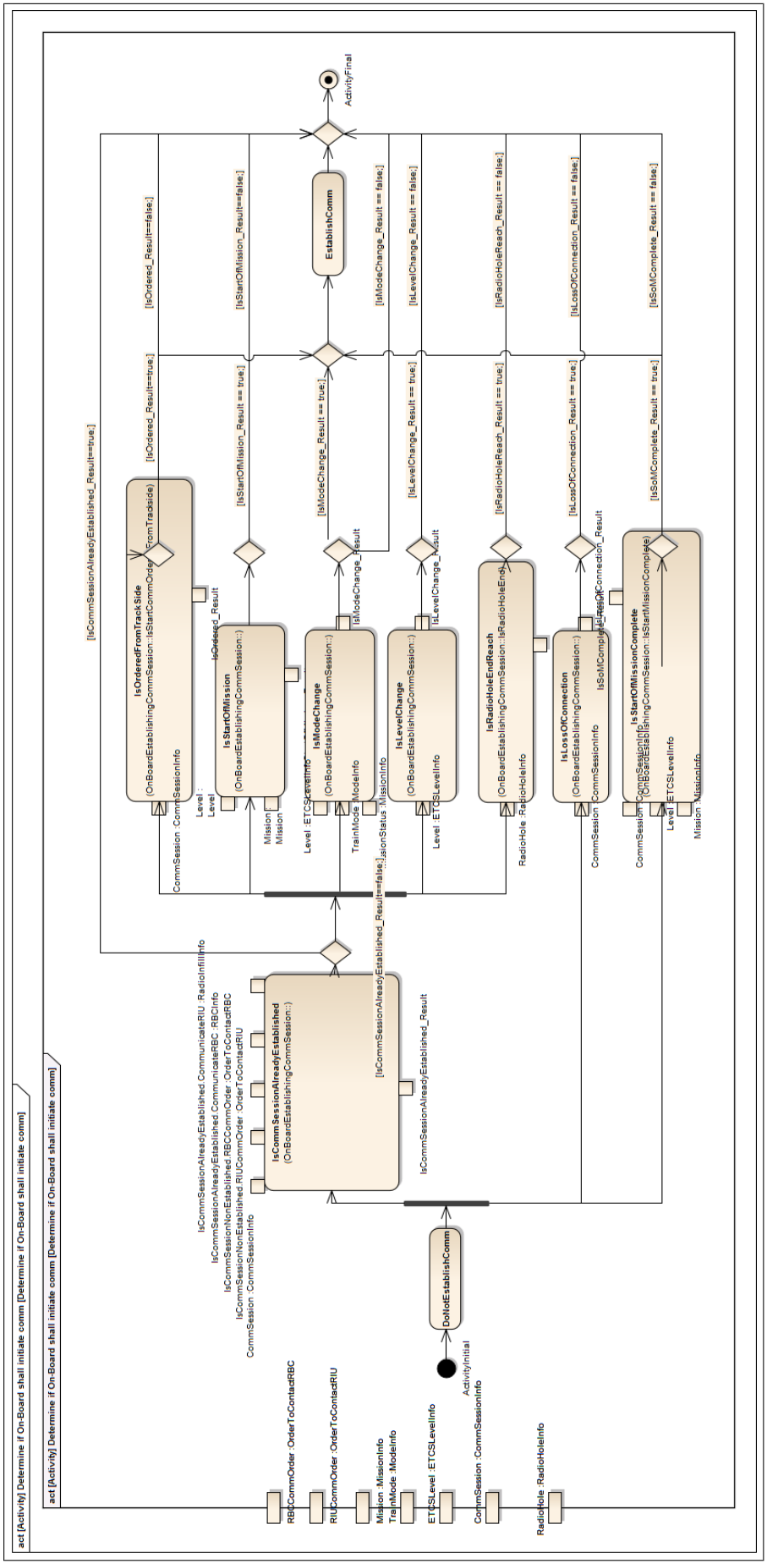


Figure 11. Activity diagram describing how the on-board determines if it may establish a communication session

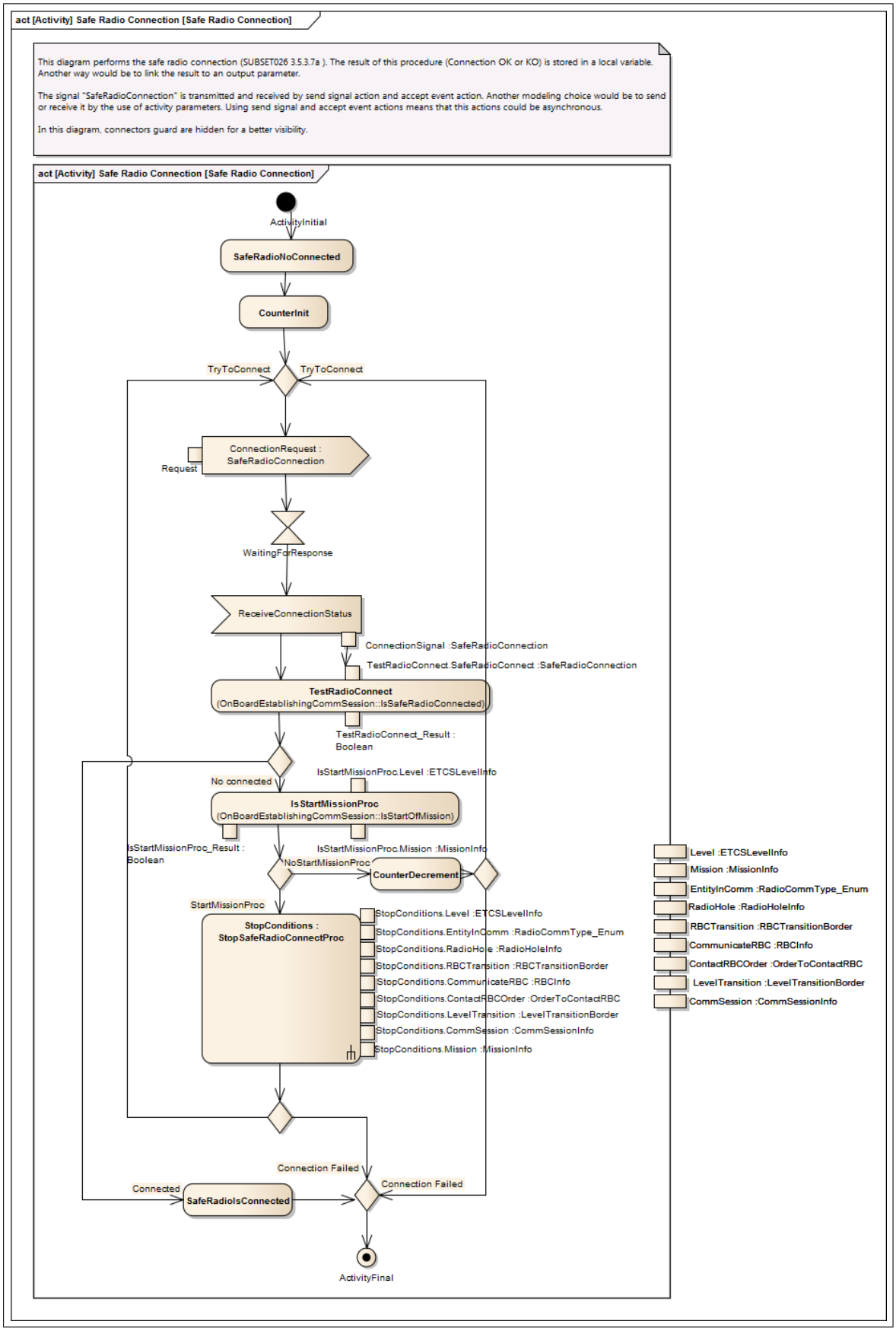


Figure 12. Activity diagram describing how the on-board shall establish a safe radio connection

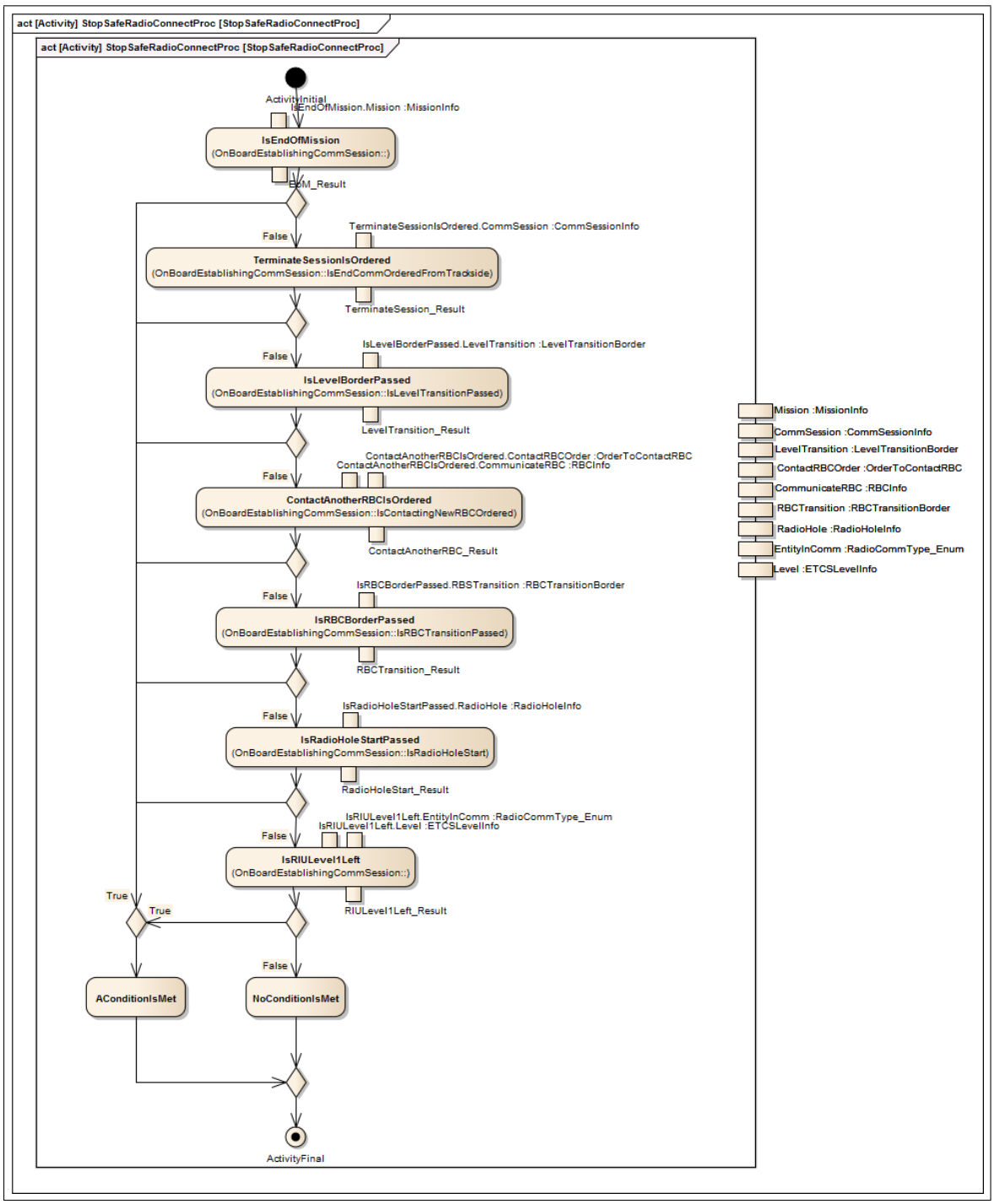


Figure 13. Activity diagram describing how the on-board shall abort the safe radio connection process

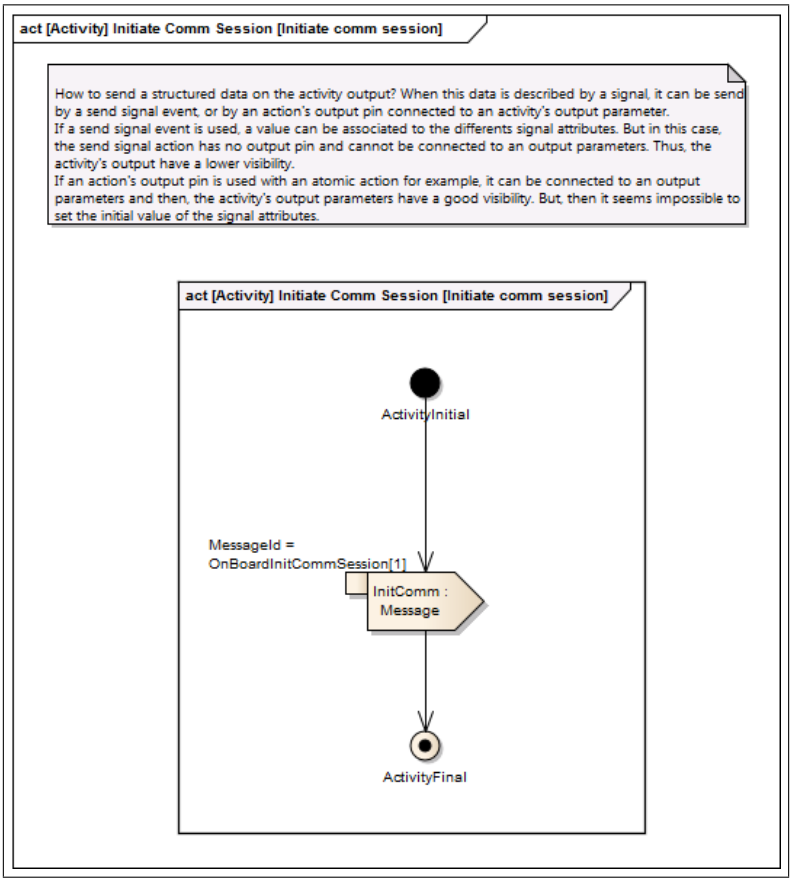


Figure 14. Activity diagram describing the initialization of a communication session by the on-board

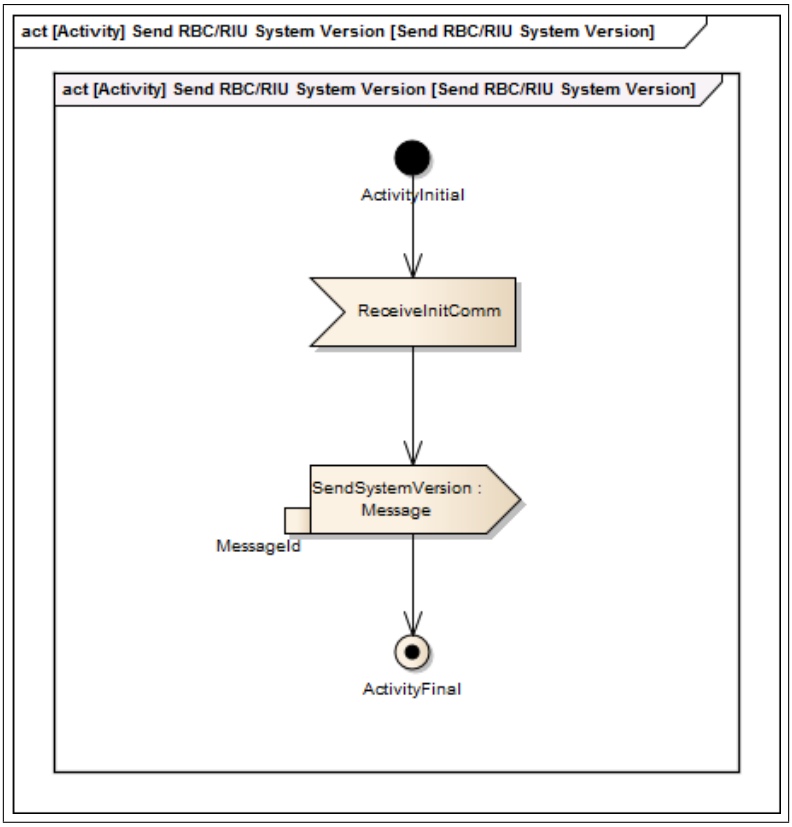


Figure 15. Activity diagram describing the trackside response to an initialization of communication session by the on-board

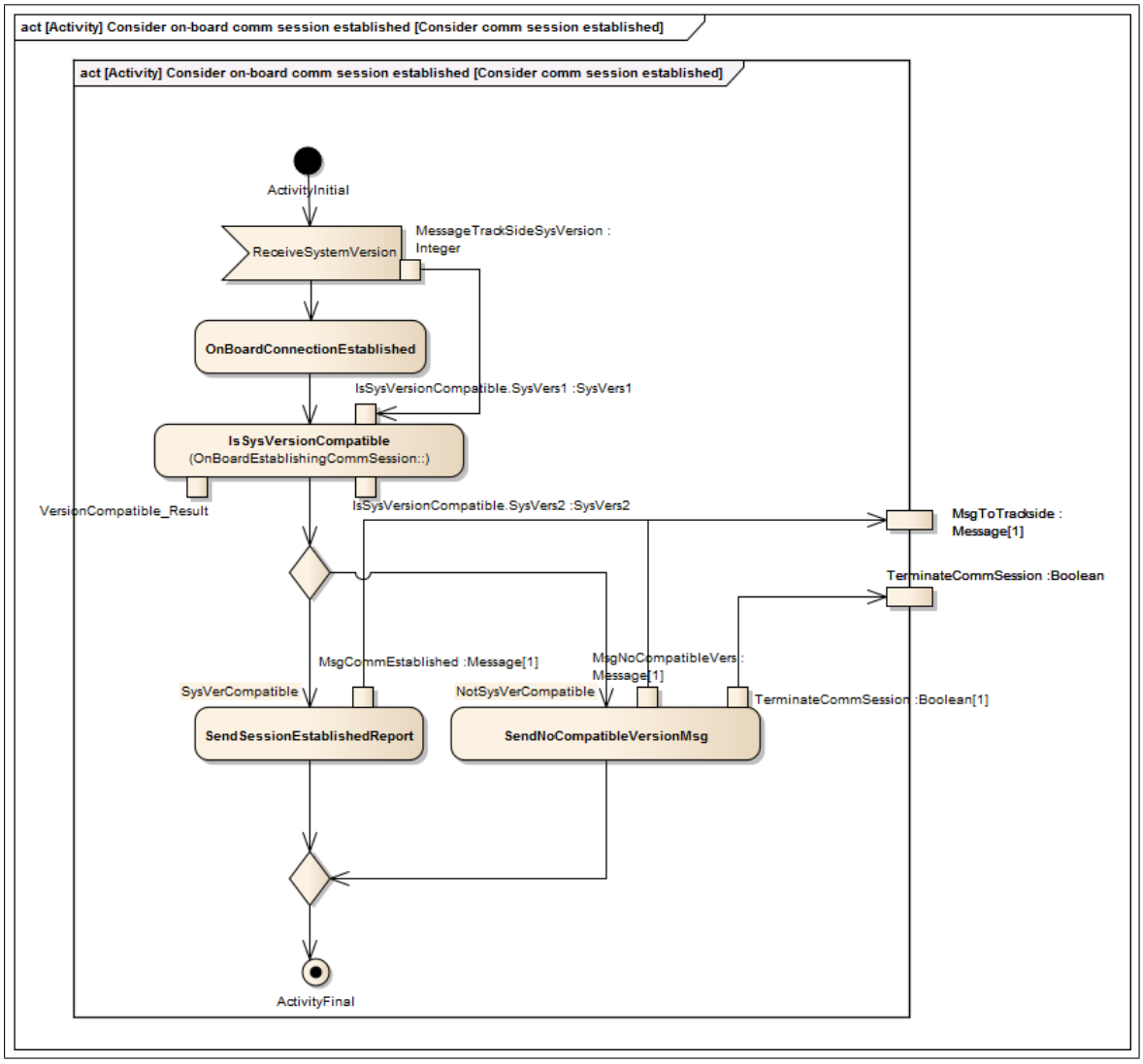


Figure 16. Activity diagram describing the on-board behavior when it receives the RBC/RIU system version

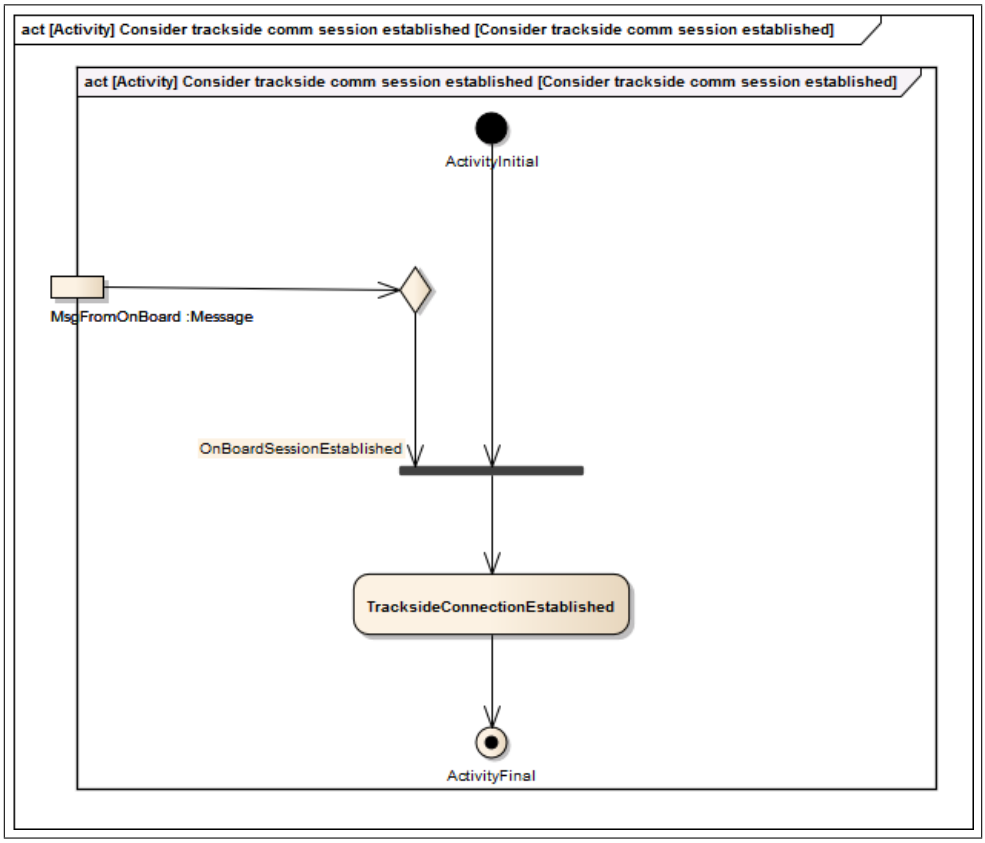


Figure 17. Activity diagram describing the trackside behavior when communication session is established by the on-board

References

[1] C. Braunstein. Radio Communication Management, sysml model. mar 2013. SUBSET026 v3.3.0.

[2] ERA. System Requirements Specification. (07), mar 2012. SUBSET026 v3.3.0.

[3] R. S. Sanford Friedenthal, Alan Moore. *A Practical Guide to SysML.* Elsevier Science, sept 2008.