Event-B Model of Subset 026, Section 3.5.3

Matthias Güdemann
Systerel, France

This document describes a formal model of the requirements of section 3.5.3 of the subset 026 of the ETCS specification 3.3.0 [?]. This section describes the establishing of a communication connection between on-board and on-track equipment.

The model is expressed in the formal language Event-B [?] and developed within the Rodin tool [?]. This formalism allows an iterative modeling approach. In general, one starts with a very abstract description of the basic functionality and step-wise adds additional details until the desired level of accuracy of the model is reached. Rodin provides the necessary proof support to ensure the correctness of the refined behavior.

In this document we present an Event-B model of the protocol to initiate a communication session in an ETCS implementation, as implemented from the on-board unit. At first, we describe shortly the background of Event-B, then the overall approach taken to model this section and finally present the model in detail. For each of the iterative modeling steps, we describe the details added by the refinement.

OBU	on board unit
RIU	radio in-fill unit
RBC	radio block centre
SRS	system requirements specification

Table 1: Glossary

1 Short Introduction to Event-B

The formal language Event-B is based on a set-theoretic approach. It is a variant of the B language, with a focus on system level modeling [?]. An Event-B model is separated into a static and a dynamic part.

The dynamic part of an Event-B model describes abstract state machines. The state is represented by a set of state variables. A transition from one state to another is represented by parametrized events which assign new values to the state variables. Event-B allows unbounded state spaces. They are constrained by invariants expressed in first order logic with equality which must be fulfilled in any case. The initial state is created by a special initialization event.

The static part of an Event-B model is represented by contexts. These consist of carrier sets, constants and axioms. The type system of a model is described by means of carrier sets and constraints expressed by axioms.

Event-B is not only comprised of descriptions of abstract state machines and contexts, but also includes a development approach. This approach consists of iterative refinement of the machines until the desired level of detail is reached. In the Rodin tool, proof obligations are automatically created which ensure correct refinement.

Together with the machine invariants, the proof obligations for the refinement are formally proven, creating proof trees. To accomplish this, there are different options: many proof obligations can be discharged by automated provers (e.g., AtelierB, NewPP, Rodin's SMT-plugin), but as the underlying logic is in general undecidable, it is sometimes necessary to use the interactive proof support of Rodin.

Any external actions, e.g., mode changes by the driver or train level changes are modeled via parametrized events. Only events can modify the variables of a machine. An Event-B model is on the system level, events are assumed to be called from a software system into which the functional model is embedded. The guards of the events assure that any event can only be called when appropriate.

2 Modeling Strategy

The section 3.5.3 of the SRS describes how a communication session is established. In its context, the low level EURORADIO network connection (cf. §3.5.1.1) is considered basic functionality and is not part of the modeling.

The basic modeling element are entities which represent one piece of equipment, either on-board, i.e., on the train, or on-track. The model is constructed from the local point of view of an OBU entity. On-track entities are only modeled as possible communication partners.

3 Model Overview

Figure 1 shows the structure of the Event-B model. The left column represents the abstract state machines, the right column the contexts. An arrow from one machine to another machine represents a refinement relation, an arrow from a machine to a context represents a sees relation and arrow from one context to another represents an extension relation.

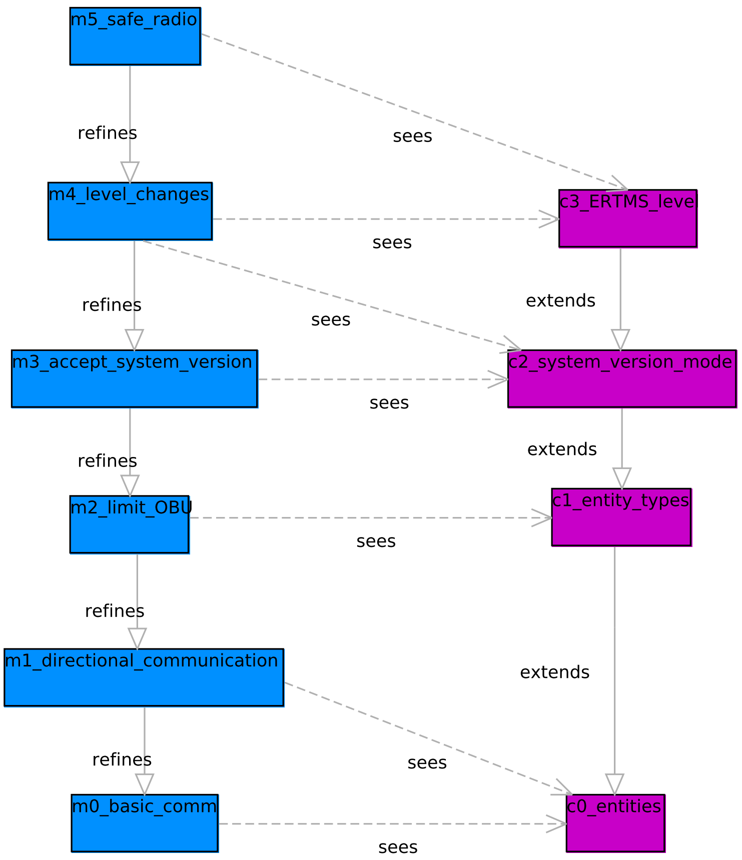

Figure 1: Overview on State Machine and Context Hierarchy

The modeling starts with the very abstract possibility to establish and to terminate a communication session in the machine $m0$, the set of entities is defined in the context $c0$. This basic functionality is refined in the succeeding machines to incorporate the different stages of the protocol to establish a session. The contexts further refine the entities to on-track and on-board entities and limit the modeling to the point of view of an OBU.

The machine $m1$ discerns incoming and outgoing communication sessions, i.e., initiated by the modeled piece of equipment or by an external one. The context $c1$ introduces the different types of equipment which are used in $m2$ to refine the two different protocols for outgoing and incoming sessions and to limit the model to the OBU point of view. $c2$ introduces the notion of compatible systems. This is used in $m3$ to identify on-track equipment with a compatible system version. This machine also discerns between accepting and non-accepting RBCs to contact. $c3$ adds the different ERTMS levels and the relevant train modes to the model. This is used in $m4$ to model the different situations where a communication session must be established. $m5$ adds the notion of safe radio connection as low-level prerequisite for a communication session.

The representation of the state machines of the modeled protocols for establishing a communication session is modeled implicitly. The model allows sessions with different partners in parallel (but respects the constraints of the specification like §3.5.3.5.2). The state of the protocol with different partners is tracked by adding / removing these partners from sets representing those different states of the protocol.

4 Model Benefits

The Event-B model in Rodin has some interesting properties which are highlighted here. Some stem from the fact that Rodin is well integrated into the Eclipse platform which renders many useful plugins available, both those explicitly developed for integration with Rodin, but also other without Rodin in mind. Other interesting properties stem from the fact that Rodin and Event-B provide an extensive proof support for properties.

- **Refinement** The Event-B approach allows iterative development based on refinement. This allows starting modeling with a very abstract machine and then step-wise adding more detailed behavior. Rodin generates all the necessary proof obligations which are required to assure correct refinement.
- **Requirements Tracing** Rodin provides an extensible EMF model, therefore it is easily possible to trace requirements using the requirements modeling framework of Eclipse (RMF) via the ProR plugin. This allows the usage of requirement documents in the OMG standardized Requirements Interchange Format (ReqIF).
- **Model Animation** The Event-B model can be animated via different plugins, e.g., ProB or AnimB. This allows the simulation of the model, by clicking on the activated events and tracking the resulting state of the variables. This technique allows to examine the run-time behavior of the model, e.g., for testing purposes. There is also ongoing development for a model-based testing plugin in Rodin, which will allow storing and replaying of event sequences.
- **Non-Testable Requirements** The Event-B model supports the specification of invariants which can be formally proven using the proof support of Rodin. This includes for example the non-testable requirement specified in the subset 076 for §3.5.3.2 (see Section 5.5).
- **Safety Properties** Using Rodin's proof support and the formalization as invariants, it is possible to formalize and prove the identified safety property of the case study (see Section 5.7).

5 Detailed Model Description

This section describes in more detail the formal model, beginning from the most abstract Event-B machine. For each refinement, in general only the important changes will be shown, the complete model is available as a Rodin project. At each step the additional modeled functionality and its representation will be described. In particular the initialization event is not shown for the refined machines. If not mentioned explicitly, sets are initialized empty, integers with value 0 and Boolean variables with false.

5.1 Context 0 - Entities

This context defines the type of entities with whom a communication session can be established. *my_entity* represents the piece of equipment which is modeled.

CONTEXT c0_entities
SETS

entities

CONSTANTS

my_entity

AXIOMS

axm1 : $my\_entity \in entities$

END

5.2 Machine 0 - Basic Communication

This state machine represents the basic functionality. It allows for the creation and the termination of a communication session with another entity. The sessions are represented by the state variable "session" which can contain values of type "entities". The respective events are triggered with a parameter $l\_partner$ representing the communication partner.

Implemented Requirements

- each session allows for communication between two entities (cf. §3.5.2.1)

SEES c0_entities

VARIABLES

$sessions$

INVARIANTS

inv1 : $sessions \subseteq entities \setminus \{my\_entity\}$

EVENTS

Initialisation

```
    begin
        act1 : $sessions := \varnothing$
    end
```

Event $establish\_communication \mathrel{\widehat{=}}$

```
    any
        $l\_partner$
    where
        grd1 : $l\_partner \notin sessions$
        grd2 : $l\_partner \neq my\_entity$
    then
        act1 : $sessions := sessions \cup \{l\_partner\}$
    end
```

Event $terminate\_communication \mathrel{\widehat{=}}$

```
    any
        $l\_partner$
    where
        grd1 : $l\_partner \in sessions$
    then
        act1 : $sessions := sessions \setminus \{l\_partner\}$
    end
```

END

5.3 Machine 1 - Directional Communication

The first refinement of the machine refines the notion of communication session to incoming sessions, i.e., where another entity initiates a session with *my_entity* and outgoing sessions where *my_entity* initiates the session.

The data refinement is proven by the invariant which states that "sessions" is equal to the disjoint union of "outgoing_sessions" and "incoming_sessions". The abstract "establish_session" event is refined to the two events "incoming_session" and "outgoing_session".

REFINES m0_basic_comm
SEES c0_entities
VARIABLES

  $incoming\_sessions$
  $outgoing\_sessions$

INVARIANTS

  inv1 : $partition(sessions, incoming\_sessions, outgoing\_sessions)$

EVENTS
Event $incoming\_communication \mathrel{\widehat{=}}$
refines $establish\_communication$
  any
    $l\_partner$
  where
    grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
    grd2 : $l\_partner \neq my\_entity$
  then
    act1 : $incoming\_sessions := incoming\_sessions \cup \{l\_partner\}$
  end
Event $outgoing\_communciation \mathrel{\widehat{=}}$
refines $establish\_communication$
  any
    $l\_partner$
  where
    grd2 : $l\_partner \neq my\_entity$
    grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
  then
    act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
  end
Event $terminate\_communication \mathrel{\widehat{=}}$
refines $terminate\_communication$
  any
    $l\_partner$
  where
    grd1 : $l\_partner \in incoming\_sessions \cup outgoing\_sessions$
  then
    act1 : $incoming\_sessions := incoming\_sessions \setminus \{l\_partner\}$
    act2 : $outgoing\_sessions := outgoing\_sessions \setminus \{l\_partner\}$
  end
END

5.4 Context 1 - Entity Types

The first context extension introduces the different types of entities relevant in this requirement subset, i.e., on-board unit (OBU), radio in-fill unit (RIU) or radio block centre (RBC). Every entity has a unique type. The goal is to model the communication protocol from the point of view of an OBU, therefore the type of *my_entity* is restricted to OBU.

CONTEXT c1_entity_types
EXTENDS c0_entities
CONSTANTS

> *RBC*
> *RIU*
> *OBU*
> *on_track*
> *on_board*

AXIOMS

> axm1 : $partition(entities, RBC, RIU, OBU)$
> axm2 : $on\_track = RIU \cup RBC$
> axm3 : $on\_board = OBU$
> axm4 : $my\_entity \in on\_board$

END

5.5 Machine 2 - On Board Modeling

The next machine refinement adds the notion of being contacted by an on-track entity to establish a communication session. It also adds the first state of the protocol, i.e., entities which are contacted with the “Initiation of a communication session” message. On-track entities which order “my_entity” to contact are stored in the “contacted_by” set, entities to which the first message is sent by “my_entity” are stored in the set “contacted’, representing those which are in the first stage of the protocol.

The invariants prove that *my_entity* will only be in contact with on-track entities and that any entities which are considered for a communication session are on-track entities. Any entity with whom there is already a communication session will not be considered for another session, and finally no radio in-fill unit can initiate a communication session with *my_entity*.

Implemented Requirements

- It shall be possible for OBU and RBC to initiate communication session (cf. §3.5.3.1)
- RIU cannot initiate a communication session (cf. §3.5.3.2)

  This invariant is marked as non-testable in Subset-076.

The other invariants ensure that a communication partner is not in different states of the communication protocol at the same time. A session protocol can be started by the order to contact an RBC or directly by the OBU.

REFINES m1_directional_communication
SEES c1_entity_types
VARIABLES

> *contacted*
> *contacted_by*

INVARIANTS

> inv1 : $incoming\_sessions \cup outgoing\_sessions \subseteq on\_track$
> inv2 : $contacted \subseteq on\_track$
> inv3 : $contacted\_by \subseteq on\_track$

inv4 : $contacted\_by \cap (incoming\_sessions \cup outgoing\_sessions) = \varnothing$
inv5 : $contacted \cap (incoming\_sessions \cup outgoing\_sessions) = \varnothing$
inv6 : $incoming\_sessions \cap RIU = \varnothing$
inv7 : $contacted \cap contacted\_by = \varnothing$

EVENTS

Event *incoming_communication* $\widehat{=}$
refines *incoming_communication*

```
    any
        l_partner
    where
        grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
        grd3 : $l\_partner \in on\_track \setminus RIU$
        grd4 : $l\_partner \notin contacted$
        grd5 : $l\_partner \notin contacted\_by$
    then
        act1 : $incoming\_sessions := incoming\_sessions \cup \{l\_partner\}$
    end
```

Event *receive_contact_order* $\widehat{=}$

```
    any
        l_partner
    where
        grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
        grd2 : $l\_partner \in on\_track$
    then
        act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
    end
```

Event *initiate_session_after_contact* $\widehat{=}$

```
    any
        l_partner
    where
        grd2 : $l\_partner \in contacted\_by$
    then
        act1 : $contacted := contacted \cup \{l\_partner\}$
        act2 : $contacted\_by := contacted\_by \setminus \{l\_partner\}$
    end
```

Event *initiate_session_no_contact* $\widehat{=}$

```
    any
        l_partner
    where
        grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
        grd3 : $l\_partner \in on\_track$
    then
        act2 : $contacted := contacted \cup \{l\_partner\}$
    end
```

END

5.6 Context 2 - System Version Compatibility

The next context extension introduces the notion of compatible system versions. This is modeled as a static property of the on-track equipment wrt. *my_entity*, i.e., the context axiom that "system_version_compatible" is a subset of all on-track entities. On this level of abstraction, there is no need for a finer grained notion of compatibility.

EXTENDS c1_entity_types
CONSTANTS

$system\_version\_compatible$

AXIOMS

axm1 : $system\_version\_compatible \subseteq on\_track$

END

5.7 Machine 3 - Accepting RBC and System Version

The next machine refines the contact order events by discerning between the orders to contact an accepting or a non-accepting RBC. The notion of being an accepting RBC is considered to be a dynamic property and therefore modeled as a variable, i.e., the set "accepting".

The "receive_contact_order" event is refined by two separate events, one for orders for an accepting RBC and one for a non-accepting RBC. The "outgoing_communication" event is refined by two events, one for a compatible system version and the other for an incompatible one.

Furthermore, a just established communication session with on-track equipment with an incompatible system version will be terminated immediately after receiving this information. This is modeled by the set "terminating_session" which holds values of type entities. Only those communication sessions in this set can be closed by the termination event.

Implemented Requirements

- In case of a non-accepting RBC, all existing communication sessions with other RBCs must be terminated (cf. §3.5.3.5.2)
- After the system version is received by the OBU, the communication session is considered established and (cf. §3.5.3.8)
  - if the system version is compatible, the OBU shall send the session established message to track-side (cf. 3.5.3.8.a)
  - if the system version is incompatible, the OBU shall terminate the session (cf. 3.5.3.8.b)
- Any RBC which is contacted and with whom a communication session is established has a compatible system version (safety requirement from requirements document).

REFINES m2_limit_OBU
SEES c2_system_version_mode
VARIABLES

$terminating\_sessions$
$accepting$

INVARIANTS

inv2 : $RBC \cap outgoing\_sessions \setminus terminating\_sessions \subseteq system\_version\_compatible$
inv3 : $accepting \subseteq RBC$
inv4 : $terminating\_sessions \subseteq on\_track$

EVENTS
Event $receive\_information\_compatible \;\widehat{=}$
extends $outgoing\_communciation$
any

```
      $l\_partner$
   where
      grd3 : $l\_partner \in contacted$
      grd4 : $l\_partner \in system\_version\_compatible$
   then
      act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
      act2 : $contacted := contacted \setminus \{l\_partner\}$
   end
Event receive_information_incompatible $\widehat{=}$
extends outgoing_communciation
   any
      $l\_partner$
   where
      grd3 : $l\_partner \in contacted$
      grd4 : $l\_partner \notin system\_version\_compatible$
   then
      act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
      act2 : $contacted := contacted \setminus \{l\_partner\}$
      act3 : $terminating\_sessions := terminating\_sessions \cup \{l\_partner\}$
   end
Event receive_contact_order_accept $\widehat{=}$
refines receive_contact_order
   any
      $l\_partner$
   where
      grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
      grd3 : $l\_partner \in RIU \cup (RBC \cap accepting)$
   then
      act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
   end
Event receive_contact_order_non_accept $\widehat{=}$
refines receive_contact_order
   any
      $l\_partner$
   where
      grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
      grd3 : $l\_partner \in RIU \cup (RBC \setminus accepting)$
   then
      act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
      act2 : $terminating\_sessions := terminating\_sessions \cup (RBC \cap (incoming\_sessions \cup$
      $outgoing\_sessions))$
   end
Event terminate_communication $\widehat{=}$
extends terminate_communication
   any
      $l\_partner$
   where
      grd1 : $l\_partner \in incoming\_sessions \cup outgoing\_sessions$
      grd2 : $l\_partner \in terminating\_sessions$
   then
```

act1 : $incoming\_sessions := incoming\_sessions \setminus \{l\_partner\}$
act2 : $outgoing\_sessions := outgoing\_sessions \setminus \{l\_partner\}$
act3 : $terminating\_sessions := terminating\_sessions \setminus \{l\_partner\}$

end

Event $make\_RBC\_accepting \mathrel{\widehat{=}}$

any

$l\_partner$

where

grd1 : $l\_partner \in RBC$

then

act1 : $accepting := accepting \cup \{l\_partner\}$

end

Event $make\_RBC\_non\_accepting \mathrel{\widehat{=}}$

any

$l\_partner$

where

grd1 : $l\_partner \in accepting$

then

act1 : $accepting := accepting \setminus \{l\_partner\}$

end

END

5.8 Context 3 - ERTMS Levels

The third context introduces the notion of the different ERTMS and the notion of the mission status of a train. The modeled statuses are start of mission (SOM), end of mission (EOM) and the abstract notion of within a mission (MIS), i.e., anything between start and end of the current train mission. At this level of refinement, a more detailed modeling is not necessary.

CONTEXT c3_ERTMS_level

SETS

$ERTMS\_level$
$train\_status$

CONSTANTS

NTC
$L0$
$L1$
$L2$
$L3$
SOM start of mission
EOM end of mission
MIS while mission

AXIOMS

axm1 : $partition(ERTMS\_level, \{NTC, L0, L1, L2, L3\})$
axm2 : $partition(train\_status, \{SOM, EOM, MIS\})$

END

5.9 Machine 4 - ERTMS Level Changes

The next refined machine implements the different causes which can trigger the establishing of a communication session. The corresponding events refine the abstract “initiate_session_no_contact” event. For this, the current ERTMS level of the train is tracked, as well as its current mission status.

The indication of a level change, a mission status change, a manual level change and an announced radio hole is modeled by events. These events modify the corresponding indicator variables to signal a change and they modify the corresponding state variables.

This can be illustrated using the manual level change event as example: the Boolean variable “signal_manual_level_change” indicates that the driver manually changed the ERTMS level. It is changed by the “manual_change_level” event which is parametrized with the new level and which also modifies the “current_level” variable which models the current ERTMS level. If the new level is 2 or 3, then the train is required to establish a communication session with trackside. This is realized in the “initiate_session_no_contact_manual_change” event which reset the indication variable once the entity has been contacted. Similar events model the initiation because of non-manual level change, mission status change and announced radio holes.

Implemented Requirements

- The on-board shall establish a communication session (cf. §3.5.3.4)
  - at start of mission (only if level 2 or 3) (cf. §3.5.3.4.a)
  - if ordered from trackside (cf. §3.5.3.4.b)
  - If a mode change, not considered as an End of Mission, has to be reported to the RBC (only if level 2 or 3) (cf. §3.5.3.4.c)
  - If the driver has manually changed the level to 2 or 3 (cf. §3.5.3.4.d)
  - When the train front reaches the end of an announced radio hole (cf. §3.5.3.4.e)

REFINES m3_accept_system_version
SEES c3_ERTMS_level
VARIABLES

$current\_level$
$signal\_level\_change$
$current\_status$
$signal\_status\_change$
$signal\_manual\_level\_change$
$position\_radio\_hole$
$signal\_radio\_hole$

INVARIANTS

inv1 : $current\_level \in ERTMS\_level$
inv2 : $signal\_level\_change \in BOOL$
inv3 : $current\_status \in train\_status$
inv4 : $signal\_status\_change \in BOOL$
inv5 : $signal\_manual\_level\_change \in BOOL$
inv6 : $position\_radio\_hole \in BOOL$
inv7 : $signal\_radio\_hole \in BOOL$

EVENTS

Event $manual\_change\_level \widehat{=}$

any

l_level

where

grd1 : $l\_level \in ERTMS\_level$
grd2 : $signal\_manual\_level\_change = FALSE$

```
        grd3 : $signal\_level\_change = FALSE$
    then
        act1 : $signal\_manual\_level\_change := TRUE$
        act2 : $current\_level := l\_level$
    end
Event  $change\_level \ \widehat{=}$
    any
        $l\_level$
    where
        grd1 : $l\_level \in ERTMS\_level$
        grd2 : $signal\_manual\_level\_change = FALSE$
        grd3 : $signal\_level\_change = FALSE$
    then
        act1 : $current\_level := l\_level$
        act2 : $signal\_level\_change := TRUE$
    end
extends $initiate\_session\_no\_contact$
    any
        $l\_partner$
    where
        grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
        grd3 : $l\_partner \in on\_track$
        grd6 : $current\_status = SOM$
        grd7 : $current\_level \in \{L2, L3\}$
    then
        act2 : $contacted := contacted \cup \{l\_partner\}$
    end
Event  $initiate\_session\_no\_contact\_manual\_change \ \widehat{=}$
extends $initiate\_session\_no\_contact$
    any
        $l\_partner$
    where
        grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
        grd3 : $l\_partner \in on\_track$
        grd6 : $current\_level \in \{L2, L3\}$
        grd7 : $signal\_manual\_level\_change = TRUE$
    then
        act2 : $contacted := contacted \cup \{l\_partner\}$
        act3 : $signal\_manual\_level\_change := FALSE$
    end
END
```

5.10 Machine 5 - Safe Radio Connection

The next machine refinement specifies handling of the safe radio connection which provides the necessary means to exchange protocol messages on a higher level. The existing established safe radio connections are represented by the set "ER_connections" which holds values of type entities. Safe radio connections which must be established are modeled by the set "establish_ER_connections" while "terminated_ER_connections" holds those connections which timed-out. The indication variable "signal_RBC_border" signals the crossing of an RBC border which requires to establish a new safe radio connection with a new RBC.

Implemented Requirements

- Establish communication session after safe radio connection timeout (cf. §3.5.3.4.f)
- If the communication session is established by an OBU, it shall be preformed according to the following steps (cf. §3.5.3.7)
  - if part of ongoing start of mission procedure (cf. §3.5.3.7.a)
  - if safe radio connection is set up (cf. §3.5.3.7.i)
  - if end of mission is performed (cf. §3.5.3.7.iii)
  - train passes level transition border (cf. §3.5.3.7.iv)
  - order to establish connection with different non-accepting RBC (cf. §3.5.3.7.v)
  - train passes RBC / RBC border (cf. §3.5.3.7.vi)
  - train enters announced radio hole (cf. §3.5.3.7.vii)
  - level 1 is left (RIU only) (cf. §3.5.3.7.viii)

REFINES m4_level_changes
SEES c3_ERTMS_level
VARIABLES

$ER\_connections$
$terminated\_ER\_connections$
$establish\_ER\_connection$
$signal\_RBC\_border$

INVARIANTS

inv1 : $terminated\_ER\_connections \subseteq on\_track$
inv2 : $establish\_ER\_connection \subseteq on\_track$
inv3 : $(incoming\_sessions \cup outgoing\_sessions) \subseteq ER\_connections$
inv4 : $signal\_RBC\_border \in BOOL$

EVENTS

Event $incoming\_communication \mathrel{\widehat{=}}$
extends $incoming\_communication$

```
any
    $l\_partner$
where
    grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
    grd3 : $l\_partner \in on\_track \setminus RIU$
    grd4 : $l\_partner \notin contacted$
    grd5 : $l\_partner \notin contacted\_by$
    grd6 : $l\_partner \in ER\_connections$
then
    act1 : $incoming\_sessions := incoming\_sessions \cup \{l\_partner\}$
end
```

Event $receive\_information\_compatible \mathrel{\widehat{=}}$
extends $receive\_information\_compatible$

```
any
    $l\_partner$
where
    grd3 : $l\_partner \in contacted$
    grd4 : $l\_partner \in system\_version\_compatible$
    grd5 : $l\_partner \in ER\_connections$
then
```

act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
act2 : $contacted := contacted \setminus \{l\_partner\}$
end

Event *receive_information_incompatible* $\widehat{=}$
extends *receive_information_incompatible*
any
l_partner
where
grd3 : $l\_partner \in contacted$
grd4 : $l\_partner \notin system\_version\_compatible$
grd5 : $l\_partner \in ER\_connections$
then
act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
act2 : $contacted := contacted \setminus \{l\_partner\}$
act3 : $terminating\_sessions := terminating\_sessions \cup \{l\_partner\}$
end

Event *receive_contact_order_accept* $\widehat{=}$
order to contact a RIU or accepting RBC
extends *receive_contact_order_accept*
any
l_partner
where
grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd3 : $l\_partner \in RIU \cup (RBC \cap accepting)$
grd4 : $l\_partner \notin terminated\_ER\_connections$
then
act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
end

Event *receive_contact_order_non_accept* $\widehat{=}$
extends *receive_contact_order_non_accept*
any
l_partner
where
grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd3 : $l\_partner \in RIU \cup (RBC \setminus accepting)$
grd4 : $l\_partner \notin terminated\_ER\_connections$
then
act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
act2 : $terminating\_sessions := terminating\_sessions \cup (RBC \cap (incoming\_sessions \cup outgoing\_sessions))$
end

Event *initiate_session_after_contact* $\widehat{=}$
extends *initiate_session_after_contact*
any
l_partner
where
grd2 : $l\_partner \in contacted\_by$
grd3 : $l\_partner \notin terminated\_ER\_connections$
then
act1 : $contacted := contacted \cup \{l\_partner\}$
act2 : $contacted\_by := contacted\_by \setminus \{l\_partner\}$

act3 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$
end
Event *initiate_session_no_contact_SOM* $\widehat{=}$
extends *initiate_session_no_contact_SOM*
any
l_partner
where
grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
grd3 : $l\_partner \in on\_track$
grd6 : $current\_status = SOM$
grd7 : $current\_level \in \{L2, L3\}$
grd8 : $l\_partner \notin terminated\_ER\_connections$
then
act2 : $contacted := contacted \cup \{l\_partner\}$
act3 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$
end
Event *initiate_session_no_contact_status_change* $\widehat{=}$
extends *initiate_session_no_contact_status_change*
any
l_partner
where
grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
grd3 : $l\_partner \in on\_track$
grd6 : $current\_level \in \{L2, L3\}$
grd7 : $signal\_status\_change = TRUE$
grd8 : $current\_status \neq EOM$
grd9 : $l\_partner \notin terminated\_ER\_connections$
then
act2 : $contacted := contacted \cup \{l\_partner\}$
act3 : $signal\_status\_change := FALSE$
act4 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$
end
Event *initiate_session_no_contact_manual_change* $\widehat{=}$
extends *initiate_session_no_contact_manual_change*
any
l_partner
where
grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
grd3 : $l\_partner \in on\_track$
grd6 : $current\_level \in \{L2, L3\}$
grd7 : $signal\_manual\_level\_change = TRUE$
grd8 : $l\_partner \notin terminated\_ER\_connections$
then
act2 : $contacted := contacted \cup \{l\_partner\}$
act3 : $signal\_manual\_level\_change := FALSE$
act4 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$
end
Event *initiate_session_no_contact_leave_radio_hole* $\widehat{=}$
extends *initiate_session_no_contact_leave_radio_hole*
any
l_partner
where

grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
grd3 : $l\_partner \in on\_track$
grd6 : $position\_radio\_hole = FALSE$
grd7 : $signal\_radio\_hole = TRUE$
grd8 : $l\_partner \notin terminated\_ER\_connections$

then

act2 : $contacted := contacted \cup \{l\_partner\}$
act3 : $signal\_radio\_hole := FALSE$
act4 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$

end

Event *initiate_session_after_timeout* $\widehat{=}$
extends *initiate_session_no_contact*

any

$l\_partner$

where

grd5 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted \cup contacted\_by$
grd3 : $l\_partner \in on\_track$
grd6 : $l\_partner \in terminated\_ER\_connections$

then

act2 : $contacted := contacted \cup \{l\_partner\}$
act3 : $terminated\_ER\_connections := terminated\_ER\_connections \setminus \{l\_partner\}$
act4 : $establish\_ER\_connection := establish\_ER\_connection \cup \{l\_partner\}$

end

Event *establish_ER_connection_SOM* $\widehat{=}$

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in establish\_ER\_connection$
grd3 : $current\_status = SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$
act2 : $ER\_connections := ER\_connections \cup \{l\_partner\}$

end

Event *establish_ER_connection* $\widehat{=}$

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in establish\_ER\_connection$
grd3 : $current\_status \neq SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$
act2 : $ER\_connections := ER\_connections \cup \{l\_partner\}$

end

Event *est_perform_end_of_mission* $\widehat{=}$

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in establish\_ER\_connection$

grd3 : $signal\_status\_change = TRUE$
grd4 : $current\_status = EOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *est_pass_level_transition* $\widehat{=}$

any

l_partner

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in establish\_ER\_connection$
grd3 : $signal\_level\_change = TRUE$
grd4 : $current\_status \neq SOM$
grd5 : $current\_level \in \{L0, L1, NTC\}$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *est_pass_radio_hole* $\widehat{=}$

any

l_partner

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in establish\_ER\_connection$
grd3 : $signal\_radio\_hole = TRUE \land position\_radio\_hole = TRUE$
grd4 : $current\_status \neq SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *est_RIU_leave_L1* $\widehat{=}$

any

l_partner

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in RIU$
grd3 : $signal\_level\_change = TRUE$
grd5 : $current\_level \neq L1$
grd4 : $current\_status \neq SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *est_RBC_border* $\widehat{=}$

any

l_partner

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in RBC$
grd3 : $signal\_RBC\_border = TRUE$
grd4 : $current\_status \neq SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *indicate_RBC_border* $\widehat{=}$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_RBC\_border := l\_flag$

end

Event *est_other_RBC_non_accept* $\widehat{=}$

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$
grd2 : $l\_partner \in RBC$
grd3 : $RBC \cap accepting \cap contacted\_by \neq \varnothing$
grd4 : $current\_status \neq SOM$

then

act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$

end

Event *timeout_ER_connection* $\widehat{=}$
extends *drop_session*

any

$l\_partner$

where

grd1 : $l\_partner \in incoming\_sessions \cup outgoing\_sessions$
grd3 : $l\_partner \in ER\_connections$

then

act1 : $incoming\_sessions := incoming\_sessions \setminus \{l\_partner\}$
act2 : $outgoing\_sessions := outgoing\_sessions \setminus \{l\_partner\}$
act3 : $ER\_connections := ER\_connections \setminus \{l\_partner\}$
act4 : $terminated\_ER\_connections := terminated\_ER\_connections \cup \{l\_partner\}$

end

Event *terminate_communication* $\widehat{=}$
extends *terminate_communication*

any

$l\_partner$

where

grd1 : $l\_partner \in incoming\_sessions \cup outgoing\_sessions$
grd2 : $l\_partner \in terminating\_sessions$
grd3 : $l\_partner \notin terminated\_ER\_connections$

then

act1 : $incoming\_sessions := incoming\_sessions \setminus \{l\_partner\}$
act2 : $outgoing\_sessions := outgoing\_sessions \setminus \{l\_partner\}$
act3 : $terminating\_sessions := terminating\_sessions \setminus \{l\_partner\}$
act4 : $ER\_connections := ER\_connections \setminus \{l\_partner\}$

end

Event *make_RBC_accepting* $\widehat{=}$
extends *make_RBC_accepting*

any

$l\_partner$

where

  grd1 : $l\_partner \in RBC$

then

  act1 : $accepting := accepting \cup \{l\_partner\}$

end

Event *make_RBC_non_accepting* $\widehat{=}$
extends *make_RBC_non_accepting*

any

  l_partner

where

  grd1 : $l\_partner \in accepting$

then

  act1 : $accepting := accepting \setminus \{l\_partner\}$

end

END