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 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 in the RODIN tool [?]. In this documentation we present the Event-B machines and the contexts of the model. Each section introduces one refinement step, discusses the reasoning of that step and the newly introduced variables and invariants. Later sections refine the machine, introducing more detailed behavior.

The machines are not presented in full, only the relevant parts that were changed or added will be shown to make it more readable. In particular the initialization is not shown for the refined machines. If not mentioned explicitly, sets are initialized empty, integers with value 0 and Booleans with false.

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 development [?]. An Event-B model describes abstract state machines, transitions describe changes to the state variables of the machine and pre-conditions of the events are expressed in first order logic with equality. The Event-B machines also contain invariant specifications in this logic. These represent requirements assuring the correctness of the model.

While Event-B machines describe the dynamic aspect of a model, contexts described the static part. In particular, they describe the type system of a model by means of carrier sets. Contexts also allow the definition of constants and axioms, in general these axioms define constraints on the types.

Event-B is not only the description of an abstract state machine and its type properties, but is also comprised of a development approach. This approach consists of iterative refinement of the machines until the desired level of detail is reached. To ensure correct refinement, i.e., that the abstract model has a more general behavior, proof obligations are created. This can be automated by special tools like Rodin. For refinement of abstract data structures, the necessary proof obligations must be created manually.

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

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) are considered basic functionality and are not part of the modeling.

The model is constructed from the local point of view of an OBU entity, it does not consider modeling any on-track unit. On track entities are only modeled as possible communication partners.

Established communication sessions are modeled as sets of entities, the events that modify these sets have an event parameter that represents one entity

3 Model Overview

Figure 1 shows the structure of the Event-B model. The blue boxes represent the abstract state machines, the magenta boxes 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.

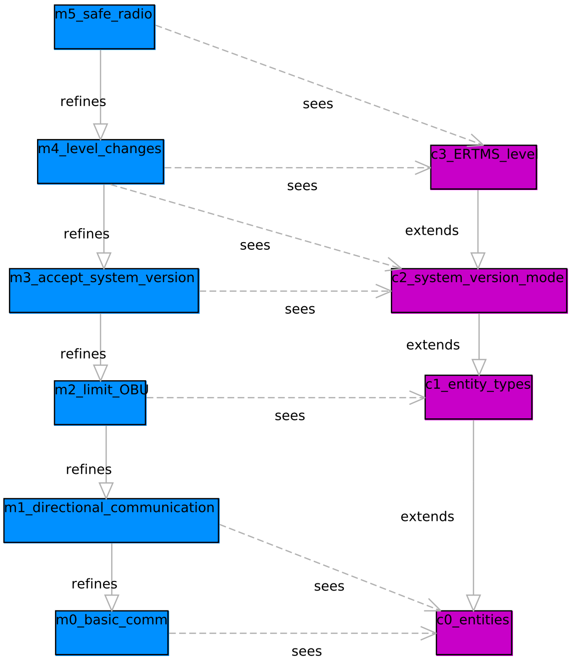

Figure 1: Overview on State Machine and Context Hierarchy

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 to the model.

All external actions, e.g., mode changes by the driver or train level changes are modeled via 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, e.g., a communication session will not be established before an EURORADIO connection exists.

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 Highlights

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 apply the defined test cases to the model.
- **Non-Testable Requirements** The Event-B model supports the specification of invariants which can be formally proven using the Rodin's proof support. This includes for example some of the non-testable requirements of Subset 076.
- **Safety Properties** Using Rodin's proof support and the formalization as invariants, it is possible to formalize and prove the identified safety properties for the case study.

5 Detailed Model Description

This section describes in detail the formal model, beginning from the most abstract Event-B machine. In general only the important changes will be shown, as the complete model is available as a Rodin project. At each step the additional modeled functionality will be described.

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 abstract state machine represents the basic functionality. Its allows for the creation and the destruction of a communication session with another entity. The respective events are triggered with a parameter *l_partner* which represents the potential communication partner.

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

MACHINE m0_basic_comm

SEES c0_entities

VARIABLES

$comm\_sessions$

INVARIANTS

inv1 : $comm\_sessions \subseteq entities \setminus \{my\_entity\}$

EVENTS

Initialisation

begin

act1 : $comm\_sessions := \varnothing$

end

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

any

$l\_partner$

where

grd1 : $l\_partner \notin comm\_sessions$

grd2 : $l\_partner \in entities \setminus \{l\_partner\}$

then

act1 : $comm\_sessions := comm\_sessions \cup \{l\_partner\}$

end

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

any

$l\_partner$

where

grd1 : $l\_partner \in comm\_sessions$

then

act1 : $comm\_sessions := comm\_sessions \setminus \{l\_partner\}$

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 establishes a session with my_entity and outgoing sessions where my_entity establishes the session.

The data refinement is proven by the invariant which state that the union of outgoing and incoming sessions is equal to the communication sessions and that the intersection of the sets of incoming and outgoing session is empty. The abstract "establish_session" event is refined to the two events "incoming_session" and "outgoing_session".

VARIABLES

$incoming\_sessions$

$outgoing\_sessions$

INVARIANTS

inv1 : $comm\_sessions = incoming\_sessions \cup outgoing\_sessions$

inv2 : $incoming\_sessions \cap outgoing\_sessions = \varnothing$

EVENTS

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

refines $establish\_communication$

any

```
        $l\_partner$
    where
        grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
        grd2 : $l\_partner \in entities \setminus \{l\_partner\}$
    then
        act1 : $incoming\_sessions := incoming\_sessions \cup \{l\_partner\}$
    end
Event outgoing_communciation $\widehat{=}$
refines establish_communication
    any
        $l\_partner$
    where
        grd2 : $l\_partner \in entities \setminus \{l\_partner\}$
        grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
    then
        act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
    end
```

5.4 Context 1 - Entity Types

The first context extension introduces the different types of entities relevant for this requirement subset, i.e., OBU, RIU, RBC. It restricts the type of *my_entity* to OBU and adds the distinction between on-track and on-board entities.

```
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 communication session establishing protocol, i.e., entities which are contacted with the "Initiation of a communication session" message.

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*.

- 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.

VARIABLES

$contacted$

$contacted\_by$

INVARIANTS

inv1 : $incoming\_sessions \cup outgoing\_sessions \subseteq on\_track$
limit model to OBU -> only on_track communication partners

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 : $incoming\_sessions \cap RIU = \varnothing$

EVENTS

Event $incoming\_communication \;\widehat{=}$

extends $incoming\_communication$

where

grd3 : $l\_partner \in on\_track \setminus RIU$

Event $outgoing\_communciation \;\widehat{=}$

extends $outgoing\_communciation$

where

grd3 : $l\_partner \in on\_track$

Event $receive\_contact\_order \;\widehat{=}$

RBC can order contact (cf. 3.5.3.4.b)

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 $initiate\_session\_after\_contact \;\widehat{=}$

(cf. 3.5.3.4 b) / (cf. 3.5.3.5.2)

any

$l\_partner$

where

grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions \cup contacted$

grd2 : $l\_partner \in contacted\_by$

then

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

act2 : $contacted\_by := contacted\_by \setminus \{l\_partner\}$

end

Event $initiate\_session\_no\_contact \;\widehat{=}$

no contact order, i.e., one ofthe other cases of 3.5.3.4

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 considered to be a static property of the on-track equipment, wrt. *my_entity*, therefore this is modeled as a context axiom.

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 non-accepting RBC. The notion of being an accepting RBC is considered to be a dynamic property and therefore modeling as a variable.

Furthermore, a just established communication session with on-track equipment with an incompatible system version will be terminated immediately after receiving this information.

- 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

$termination\_sessions$

$accepting$

INVARIANTS

inv1 : $termination\_sessions \subseteq incoming\_sessions \cup outgoing\_sessions$

only established sessions can be terminated

inv2 : $RBC \cap outgoing\_sessions \subseteq system\_version\_compatible$

Sylvain's safety property

inv3 : $accepting \subseteq RBC$
typing invariant for elements of accepting

EVENTS

Event *receive_information_compatible* $\widehat{=}$
extends *outgoing_communciation*
any
l_partner
where
grd2 : $l\_partner \in entities \setminus \{l\_partner\}$
grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
grd3 : $l\_partner \in on\_track$
grd4 : $l\_partner \in system\_version\_compatible$
then
act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
end

Event *receive_information_incompatible* $\widehat{=}$
extends *outgoing_communciation*
any
l_partner
where
grd2 : $l\_partner \in entities \setminus \{l\_partner\}$
grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$
grd3 : $l\_partner \in on\_track$
grd4 : $l\_partner \notin system\_version\_compatible$
then
act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$
act2 : $termination\_sessions := termination\_sessions \cup \{l\_partner\}$
end

Event *receive_contact_order_accept* $\widehat{=}$
order to contact a RIU or accepting RBC
extends *receive_contact_order*
any
l_partner
where
grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd2 : $l\_partner \in on\_track$
grd3 : $l\_partner \in RIU \cup (RBC \cap accepting)$
either RIU or accepting RBC
then
act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
end

Event *receive_contact_order_non_accept* $\widehat{=}$
trackside can order contact (cf. 3.5.3.4.b)
extends *receive_contact_order*
any
l_partner
where
grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd2 : $l\_partner \in on\_track$
grd3 : $l\_partner \in RIU \cup (RBC \setminus (RBC \cap accepting))$
then

act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$

act2 : $termination\_sessions := termination\_sessions \cup (RBC \cap (incoming\_sessions \cup outgoing\_sessions))$

end

Event $terminate\_communication \widehat{=}$

extends $terminate\_communication$

any

$l\_partner$

where

grd1 : $l\_partner \in incoming\_sessions \cup outgoing\_sessions$

grd2 : $l\_partner \in termination\_sessions$

then

act1 : $incoming\_sessions := incoming\_sessions \setminus \{l\_partner\}$

act2 : $outgoing\_sessions := outgoing\_sessions \setminus \{l\_partner\}$

act3 : $termination\_sessions := termination\_sessions \setminus \{l\_partner\}$

end

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

any

$l\_partner$

where

grd1 : $l\_partner \in RBC$

then

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

end

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

any

$l\_partner$

where

grd1 : $l\_partner \in accepting$

then

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

end

END

5.8 Context 3 - ERTMS Level

The third context introduces the notion of the different ERTMS level. It also introduces the notion of start of mission, end of mission and the rather abstract while mission, i.e., anything between start and end of the current train mission.

CONTEXT c3_ERTMS_level

SETS

$ERTMS\_level$

$train\_mode$

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 : $partition(train\_mode, \{SOM, EOM, MIS\})$

END

5.9 Machine 4 - ERTMS Level Changes

The next refined machine implements mainly the different causes for establishing a communication session. 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 which modify corresponding indicator variables to signal a change and the variables which represent the current state. E.g., there is a "current_mode" variable a the corresponding "signal_level_change" variable.

- 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 c2_system_version_mode, c3_ERTMS_level

VARIABLES

$current\_level$

$signal\_level\_change$

$current\_mode$

$signal\_mode\_change$

$signal\_manual\_change$

$position\_radio\_hole$

$signal\_radio\_hole$

INVARIANTS

inv1 : $current\_level \in ERTMS\_level$

inv2 : $signal\_level\_change \in BOOL$

inv3 : $current\_mode \in train\_mode$

inv4 : $signal\_mode\_change \in BOOL$

inv5 : $signal\_manual\_change \in BOOL$

inv6 : $position\_radio\_hole \in BOOL$

inv7 : $signal\_radio\_hole \in BOOL$

EVENTS

Event $change\_level \mathrel{\widehat{=}}$

any

l_level

where

grd1 : $l\_level \in ERTMS\_level$

then

act1 : $current\_level := l\_level$

end

Event $indicate\_level\_change \mathrel{\widehat{=}}$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_level\_change := l\_flag$

end

Event $initiate\_session\_no\_contact\_SOM \mathrel{\widehat{=}}$

no contact order, i.e., one ofthe other cases of 3.5.3.4

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 : $current\_mode = SOM$
grd7 : $current\_level \in \{L2, L3\}$

Event $initiate\_session\_no\_contact\_mode\_change \mathrel{\widehat{=}}$

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 : $signal\_mode\_change = TRUE$
grd8 : $current\_mode \neq EOM$

Event $initiate\_session\_no\_contact\_manual\_change \mathrel{\widehat{=}}$

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 : $signal\_manual\_change = TRUE$

Event $initiate\_session\_no\_contact\_leave\_radio\_hole \mathrel{\widehat{=}}$

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 : $position\_radio\_hole = FALSE$
grd7 : $signal\_radio\_hole = TRUE$

END

5.10 Machine 5 - Safe Radio Connection

REFINES m4_level_changes
SEES c3_ERTMS_level, c2_system_version_mode
VARIABLES

$incoming\_sessions$
$outgoing\_sessions$
$contacted$
$contacted\_by$
$termination\_sessions$
$accepting$
$current\_level$
$signal\_level\_change$
$current\_mode$
$signal\_mode\_change$
$signal\_manual\_change$
$position\_radio\_hole$
$signal\_radio\_hole$
$ER\_connections$ set of partners with established safe radi connection
$terminated\_ER\_connections$ set of ER connections with timeouts
$establish\_ER\_connection$ set of entities which whom ER connections should be established
$signal\_RBC\_border$

INVARIANTS

inv1 : $ER\_connections \subseteq on\_track$
inv2 : $terminated\_ER\_connections \subseteq on\_track$
inv3 : $establish\_ER\_connection \subseteq on\_track$
inv5 : $terminated\_ER\_connections \cap (incoming\_sessions \cup outgoing\_sessions) = \varnothing$
inv6 : $signal\_RBC\_border \in BOOL$

EVENTS

Initialisation
extended
begin

act2 : $incoming\_sessions := \varnothing$
act3 : $outgoing\_sessions := \varnothing$
act4 : $contacted := \varnothing$
act5 : $contacted\_by := \varnothing$
act6 : $termination\_sessions := \varnothing$
act7 : $accepting := \varnothing$
act8 : $current\_level := NTC$
act9 : $signal\_level\_change := FALSE$
act10 : $current\_mode := SOM$
act11 : $signal\_mode\_change := FALSE$
act12 : $signal\_manual\_change := FALSE$
act13 : $position\_radio\_hole := FALSE$
act14 : $signal\_radio\_hole := FALSE$
act15 : $ER\_connections := \varnothing$
act16 : $terminated\_ER\_connections := \varnothing$
act17 : $establish\_ER\_connection := \varnothing$
act18 : $signal\_RBC\_border := FALSE$

end

Event $change\_pos\_radio\_hole \widehat{=}$
extends $change\_pos\_radio\_hole$

any

l_hole_pos

where

grd1 : $l\_hole\_pos \in BOOL$

then

act1 : $position\_radio\_hole := l\_hole\_pos$

end

Event $indicate\_radio\_hole \mathrel{\widehat{=}}$
extends $indicate\_radio\_hole$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_radio\_hole := l\_flag$

end

Event $indicate\_manual\_change \mathrel{\widehat{=}}$
extends $indicate\_manual\_change$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_manual\_change := l\_flag$

end

Event $change\_mode \mathrel{\widehat{=}}$
extends $change\_mode$

any

l_mode

where

grd1 : $l\_mode \in train\_mode$

then

act1 : $current\_mode := l\_mode$

end

Event $indicate\_mode\_change \mathrel{\widehat{=}}$
extends $indicate\_mode\_change$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_mode\_change := l\_flag$

end

Event $change\_level \mathrel{\widehat{=}}$
extends $change\_level$

any

l_level

where

grd1 : $l\_level \in ERTMS\_level$

then

act1 : $current\_level := l\_level$

end

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

extends $indicate\_level\_change$

any

l_flag

where

grd1 : $l\_flag \in BOOL$

then

act1 : $signal\_level\_change := l\_flag$

end

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

extends $incoming\_communication$

any

$l\_partner$

where

grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$

grd2 : $l\_partner \in entities \setminus \{l\_partner\}$

grd3 : $l\_partner \in on\_track \setminus RIU$

grd4 : $l\_partner \in ER\_connections$

then

act1 : $incoming\_sessions := incoming\_sessions \cup \{l\_partner\}$

end

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

extends $receive\_information\_compatible$

any

$l\_partner$

where

grd2 : $l\_partner \in entities \setminus \{l\_partner\}$

grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$

grd3 : $l\_partner \in on\_track$

grd4 : $l\_partner \in system\_version\_compatible$

then

act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$

end

Event $receive\_information\_incompatible\ \widehat{=}$

extends $receive\_information\_incompatible$

any

$l\_partner$

where

grd2 : $l\_partner \in entities \setminus \{l\_partner\}$

grd1 : $l\_partner \notin incoming\_sessions \cup outgoing\_sessions$

grd3 : $l\_partner \in on\_track$

grd4 : $l\_partner \notin system\_version\_compatible$

then

act1 : $outgoing\_sessions := outgoing\_sessions \cup \{l\_partner\}$

act2 : $termination\_sessions := termination\_sessions \cup \{l\_partner\}$

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 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd2 : $l\_partner \in on\_track$
grd3 : $l\_partner \in RIU \cup (RBC \cap accepting)$
either RIU or accepting RBC
grd4 : $l\_partner \notin terminated\_ER\_connections$

then

act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$

end

Event $receive\_contact\_order\_non\_accept \mathrel{\widehat{=}}$

trackside can order contact (cf. 3.5.3.4.b)

extends $receive\_contact\_order\_non\_accept$

any

$l\_partner$

where

grd1 : $l\_partner \notin contacted \cup contacted\_by \cup incoming\_sessions \cup outgoing\_sessions$
grd2 : $l\_partner \in on\_track$
grd3 : $l\_partner \in RIU \cup (RBC \setminus (RBC \cap accepting))$
grd4 : $l\_partner \notin terminated\_ER\_connections$

then

act1 : $contacted\_by := contacted\_by \cup \{l\_partner\}$
act2 : $termination\_sessions := termination\_sessions \cup (RBC \cap (incoming\_sessions \cup outgoing\_sessions))$

end

Event $initiate\_session\_after\_contact \mathrel{\widehat{=}}$

(cf. 3.5.3.4 b) / (cf. 3.5.3.5.2)

extends $initiate\_session\_after\_contact$

any

$l\_partner$

where

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

then

act1 : $contacted := contacted \cup \{l\_partner\}$
act2 : $contacted\_by := contacted\_by \setminus \{l\_partner\}$

end

Event $initiate\_session\_no\_contact\_SOM \mathrel{\widehat{=}}$

no contact order, i.e., one ofthe other cases of 3.5.3.4

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 : $current\_mode = SOM$
grd7 : $current\_level \in \{L2, L3\}$
grd8 : $l\_partner \notin terminated\_ER\_connections$

then

act2 : $contacted := contacted \cup \{l\_partner\}$

end

Event *initiate_session_no_contact_mode_change* $\widehat{=}$
extends *initiate_session_no_contact_mode_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 : $signal\_mode\_change = TRUE$
grd8 : $current\_mode \neq EOM$
grd9 : $l\_partner \notin terminated\_ER\_connections$

then

act2 : $contacted := contacted \cup \{l\_partner\}$

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 : $signal\_manual\_change = TRUE$
grd8 : $l\_partner \notin terminated\_ER\_connections$

then

act2 : $contacted := contacted \cup \{l\_partner\}$

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 : $contacted := contacted \cup \{l\_partner\}$

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 : $contacted := contacted \cup \{l\_partner\}$

end

Event $establish\_ER\_connection \ \widehat{=}$

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$

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

grd3 : $current\_mode = SOM$

then

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

act2 : $ER\_connections := ER\_connections \cup \{l\_partner\}$

end

Event $est\_perform\_end\_of\_mission \ \widehat{=}$

perform EOM while establishing session

any

$l\_partner$

where

grd1 : $l\_partner \in contacted$

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

grd3 : $signal\_mode\_change = TRUE \land current\_mode = EOM$

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 : $signal\_level\_change = TRUE \land current\_level \in \{L0, L1, NTC\}$

grd4 : $current\_mode \neq SOM$

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 : $current\_mode \neq SOM$

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 : $signal\_level\_change = TRUE \wedge current\_level \neq L1$
            grd4 : $current\_mode \neq SOM$
    then
            act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$
    end
Event  $est\_RBC\_border \mathrel{\widehat{=}}$
    any
          $l\_partner$
    where
            grd1 : $l\_partner \in contacted$
            grd2 : $l\_partner \in RBC$
            grd3 : $signal\_RBC\_border = TRUE$
            grd4 : $current\_mode \neq SOM$
    then
            act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$
    end
Event  $indicate\_RBC\_border \mathrel{\widehat{=}}$
    any
          $l\_flag$
    where
            grd1 : $l\_flag \in BOOL$
    then
            act1 : $signal\_RBC\_border := l\_flag$
    end
Event  $est\_other\_RBC\_non\_accept \mathrel{\widehat{=}}$
    any
          $l\_partner$
    where
            grd1 : $l\_partner \in contacted$
            grd2 : $l\_partner \in RBC$
            grd3 : $RBC \cap accepting \cap contacted\_by \neq \varnothing$
            grd4 : $current\_mode \neq SOM$
    then
            act1 : $establish\_ER\_connection := establish\_ER\_connection \setminus \{l\_partner\}$
    end
Event  $timeout\_ER\_connection \mathrel{\widehat{=}}$
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 : $l\_partner \in termination\_sessions$
    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 : $termination\_sessions := termination\_sessions \setminus \{l\_partner\}$

  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