OETCS

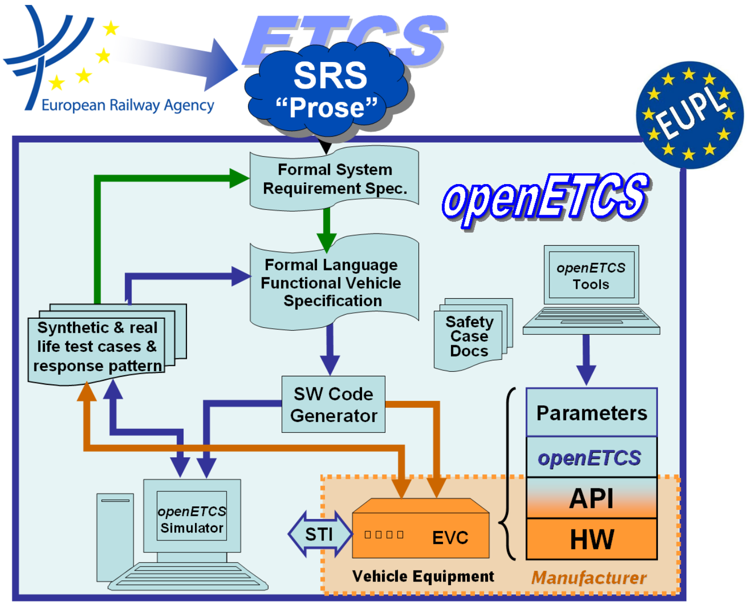
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 7: “Toolchain”

Event-B Model of Subset 026, Section 3.13

Matthias Güdemann

May 2013



Funded by:

Federal Ministry of Education and Research

Région de Bruxelles-Capitale

openETCS

This page is intentionally left blank

Work-Package 7: "Toolchain"

OETCS
May 2013

Event-B Model of Subset 026, Section 3.13

Matthias Güdemann

Systerel
Les Portes de l'Arbois, Bâtiment A
1090 rue René Descartes
13857 Aix-en-Provence Cedex 3, France

Model Description

Prepared for openETCS@ITEA2 Project

Table of Contents

1 Modeling Strategy 5
2 Model Overview 5
3 Model Benefits 7
4 Detailed Model Description 8
  4.1 Context 0 - Train Inputs, TI and DMI command 8
  4.2 Machine 0 - Train Status and Commands 9
  4.3 Machine 1 - Brake Model 10
  4.4 Context 1 - Decelerations 11
  4.5 Machine 2 - Calculate Decelerations 11
  4.6 Machine 3 - Calculation of Brake Buildup Time 12
  4.7 Machine 4 - Acceleration due to Gradient 13
  4.8 Context 3 - Speed Profiles 13
  4.9 Machine 5 - Most Restrictive Speed Profile 14
  4.10 Context 4 - Targets 14
  4.11 Machine 6 - Supervised Targets 14
  4.12 Context 5 - Braking Curves 15
  4.13 Machine 7 - Braking Curves 15
  4.14 Context 6 - Supervision Limits 16
  4.15 Machine 8 - Supervision Limit 17
5 Model Decomposition 19
  5.1 dcmp - Braking Model 19
  5.2 dcmp - Calc Deceleration 25
  5.3 dcmp - Brake Buildup 26
  5.4 dcmp - Acceleration Gradient 30
  5.5 dcmp - MRSP 34
  5.6 dcmp - Supervised Targets 35
  5.7 dcmp - Braking Curves 36
  5.8 dcmp - Supervision Limits 37
  5.9 dcmp - Monitoring Commands 38
References 41

Figures and Tables

Figures

Figure 1. Speed and Distance Monitoring Overview ([Eur12] p. 85) 6
Figure 2. Decomposition of System 7
Figure 3. Machine Decomposition Overview 7
Figure 4. Decomposition Configuration 19

Tables

This document describes a formal model of the requirements of section 3.13 of the subset 026 of the ETCS specification 3.3.0 [Eur12]. This section describes the speed and distance monitoring subsystem of ETCS.

The model is expressed in the formal language Event-B [Abr10] and developed within the Rodin tool [Jas12]. 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 speed and distance monitoring subsystem of ETCS. 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.

The section 3.13 of the SRS gives a very detailed description of the calculation of many necessary values for speed and distance monitoring. As Event-B is a system modeling approach, we give an abstract model of the system. The calculations are abstracted as functions and the system ensures the correct parameter flow to the functions. We illustrate the model decomposition capabilities of Event-B and Rodin by decomposing the overall model into different functional parts.

For a short introduction on Event-B and the usage of Rodin with models on github see `https://github.com/openETCS/model-evaluation/blob/master/model/B-Systerel/Event_B/rodin-projects-github.pdf?raw=true`

1 Modeling Strategy

The section 3.13 of the SRS describes the speed and distance monitoring together with the necessary parameters and data. The model starts with an abstract modeling of dataflow of the various intermediate calculated values. This model is partitioned into functional parts, the model is decomposed using shared variables and the respective sub-models are refined until the basic calculation functions are reached.

2 Model Overview

The overview of the speed and distance monitoring is shown in Fig. 1 from the SRS.

The on-board system comprises only the middle layer. The upper layer gives train related inputs as parameters, the lower layer track related inputs. The system itself takes the current position, speed and acceleration of the train and computes commands for the train interface and for the driver machine interface. For the train interface, this consists of the command for the service and emergency brakes. For the driver machine interface this consists of the status indication for the driver.

The Event-B modeling starts with machines describing the dataflow of all inputs, outputs and intermediate values of the model. For example, the values that are calculated for *T_brake_service* in *Traction / Braking Models* are written into a variable by an event that calculates then and these values and are read as input by the event that calculates *T_bs* for *SBI* limit.

This approach is conducted for each intermediate value of the system until a single machine is created with one variable for each intermediate value as well as for each input and output. On

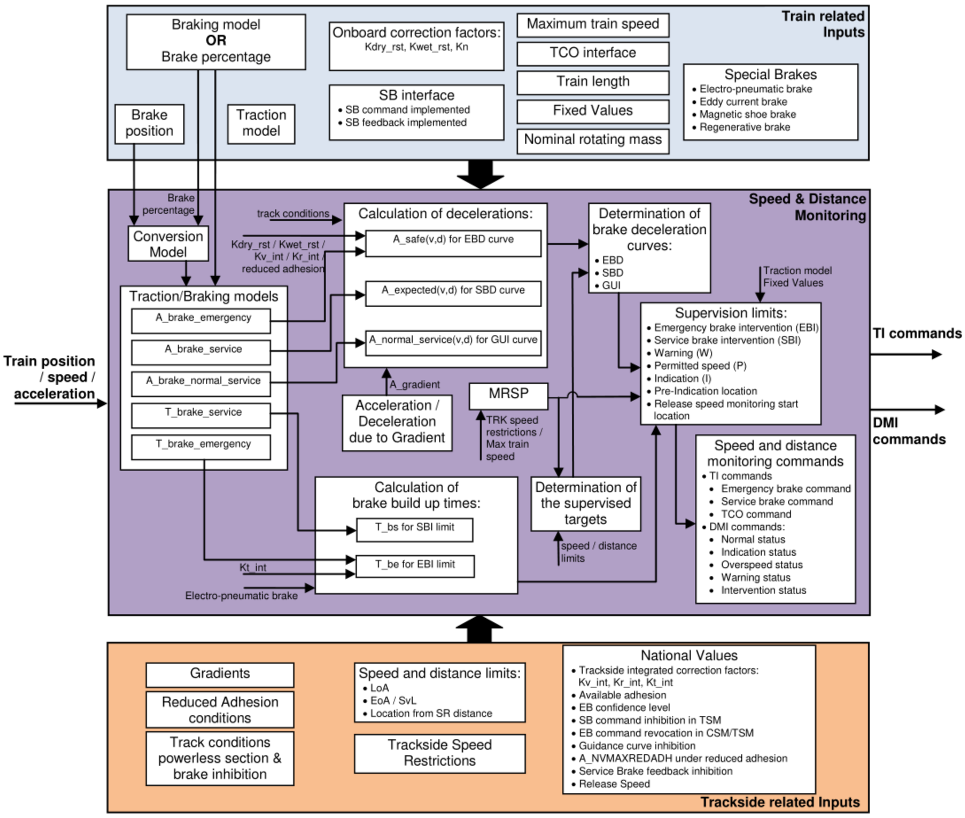


Figure 1. Speed and Distance Monitoring Overview ([Eur12] p. 85)

this level of modeling, all events only define the necessary input values and write a new value to their output variable. This value is provided as event parameter on this abstraction level.

The next step is to decompose the single machine into different sub-machines, in general one machine for each functional part of the model. This allows for model structuring and complexity reduction for each machine. For this we use the Rodin decomposition plug-in [1] using the shared-variable decomposition approach [SPHB11]. This approach splits the set of events of a machine into several disjoint sets and assigns one such set to each sub-machine. It also allows to distribute the variables over several machines, effectively implementing a shared variable distributed system.

The borders for the subsystem decomposition are shown in Fig. 2. The dashed lines show the separate sub-machines. The dataflows that cross these lines are represented by the shared variables of the decomposed model.

Each of the sub-machines with its shared variables is then further refined until the desired level of detail is reached. The overview of these refinements is shown in Fig. 3.

This refinement and context overview is very different from the others, as first an abstract global model was developed and then this model was decomposed into sub-models which are further refined. The contexts are shared between the decomposed models as far as possible. In this case, all resulting contexts and machines are kept in the same Rodin project. It is also possible to create a new project for each sub-machine which will reduce the complexity of each single project.

[1] `http://wiki.event-b.org/index.php/Decomposition_Plug-in_User_Guide`

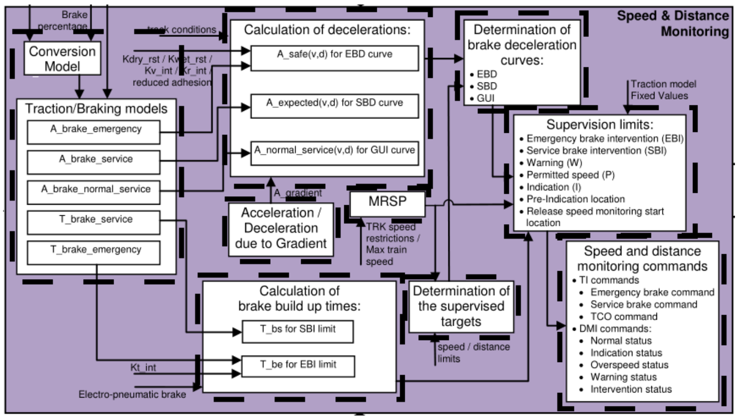


Figure 2. Decomposition of System

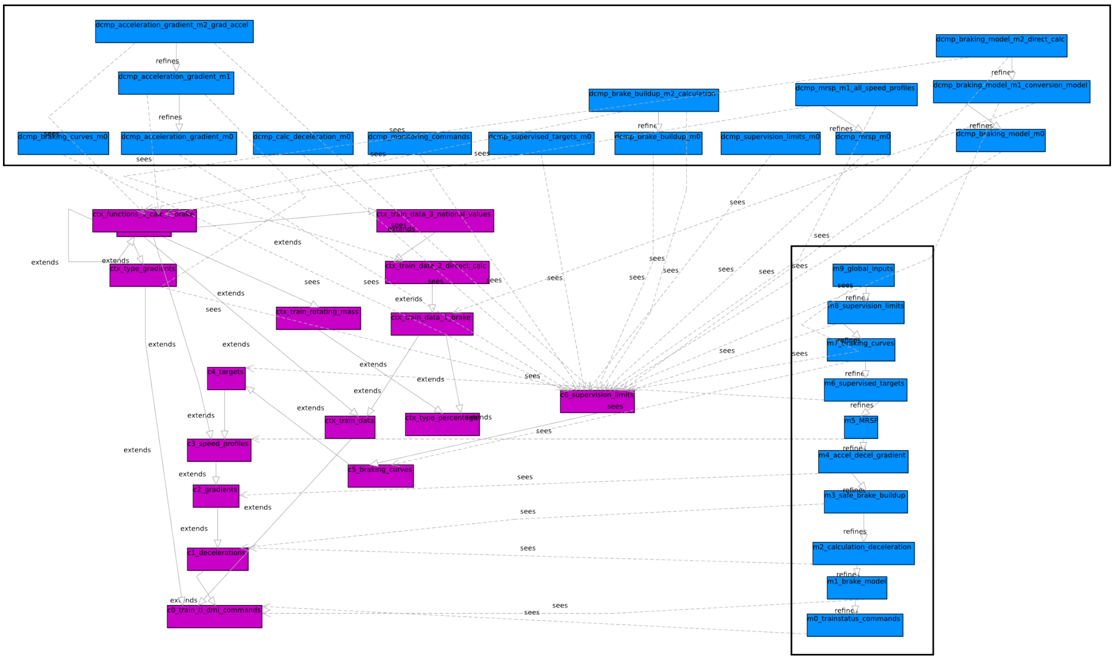


Figure 3. Machine Decomposition Overview

The global model is shown in the lower right. The first machine describes the global input and output variables of the system. The further refinements represent the iterative addition of more functions as shown in Fig. 2. For example the machine 1 adds the brake model with its inputs and outputs and the machine 2 adds the calculation of deceleration which uses the outputs of the braking model.

The last machine is then decomposed into the nine machines representing each a single functional block. This structure is shown in the upper part of Fig. 3, this also illustrates the further independent refinement of the decomposed sub-machine.

The context hierarchy also reflects this structuring. The contexts define the data types for the intermediate values, as well as the functions that calculate these values. These functions are generally not further refined in the Event-B model, as this is not part of the system modeling.

3 Model Benefits

The modeled section of the SRS provides many details for calculation of various values. The main content from a system modeling point of view is the model. So while in this case the same benefits from using Rodin as for [Mat13a, Mat13b, Mat13c] are present, the main advantage here is the model structuring facility.

- **Model Decomposition** The shared variable model decomposition [SPHB11] allows for decomposing an Event-B model and for separate refining of the machines of the resulting sub-models while retaining correctness of the refinement proofs.

It should be noted that this section contains mainly very specific implementation details and no general requirements. Currently, the proof support for non-linear arithmetic (in particular for floating point numbers) is limited in Rodin, so the modeling contains mainly the system level. It describes in particular the decomposition of the model, the various inputs and outputs of the different model components and the refinement stops when the functional level is reached. This means that for calculations, the last refinement level in general describes a function with the required input and output value and correct types.

4 Detailed Model Description

4.1 Context 0 - Train Inputs, TI and DMI command

The first context introduces many basic type for the model, *t_locations*, *t_speed*, *t_acceleration*, *t_TI_commands*, *t_DMI_commands*, *t_time* and *t_train_modes*.

The commands for the train interface (TI) are represented by the constants *c_emergency_brake*, *c_service_brake*, *c_TCO*, *c_no_command*. For the driver machine interface (DMI) the commands are represented by the constants *c_normal*, *c_indication*, *c_overspeed*, *c_warning* and *c_intervention*.

The other constants provide default values for the initialization of variables of that type.

CONTEXT c0_train_ti_dmi_commands
SETS

t_locations all possible locations on track
t_speed train speed measurement
t_acceleration train acceleration
t_TI_commands track interface commands
t_DMI_commands driver machine interface commands
t_time
t_train_modes

CONSTANTS

c_emergency_brake
c_service_brake
c_TCO traction cut off
c_no_command empty command
c_normal
c_indication
c_overspeed

$c\_warning$
$c\_intervention$
c_v0
c_a0
c_l0
c_a_brake0
c_T_brake0

AXIOMS

axm1 : $partition(t\_TI\_commands, \{c\_no\_command\}, \{c\_emergency\_brake\}, \{c\_service\_brake\}, \{c\_TCO\})$

axm2 : $partition(t\_DMI\_commands, \{c\_normal\}, \{c\_indication\}, \{c\_overspeed\}, \{c\_warning\}, \{c\_intervention\})$
axm3 : $c\_v0 \in t\_speed$
axm4 : $c\_a0 \in t\_acceleration$

axm5 : $c\_l0 \in t\_locations$
axm6 : $c\_a\_brake0 \in t\_speed \rightarrow t\_acceleration$
default brake profile

axm7 : $c\_T\_brake0 \in t\_time$
default brake buildup time

END

4.2 Machine 0 - Train Status and Commands

This first machine introduces the external input variables, i.e., the position, speed and acceleration of the train as well as the output variables, i.e., the TI commands and the DMI commands. The input variables are read by the event *update_train_style* and the output variables by the event *new_outputs*.

MACHINE m0_trainstatus_commands
SEES c0_train_ti_dmi_commands
VARIABLES

$v\_current$ current speed of train
$a\_current$ current acceleration of train
$loc\_current$ current position of train as track location
$cmd\_current$ current TI command
$status\_current$ current DMI status

INVARIANTS

inv1 : $v\_current \in t\_speed$
inv2 : $a\_current \in t\_acceleration$
inv3 : $loc\_current \in t\_locations$
inv4 : $cmd\_current \in t\_TI\_commands$
inv5 : $status\_current \in t\_DMI\_commands$

EVENTS
Initialisation
begin

act1 : $v\_current := c\_v0$
act2 : $a\_current := c\_a0$
act3 : $loc\_current := c\_l0$
act4 : $cmd\_current := c\_no\_command$
act5 : $status\_current := c\_normal$

end

Event *update_train_state* $\widehat{=}$

any

l_speed
l_accel
l_loc

where

grd1 : $l\_speed \in t\_speed$
grd2 : $l\_accel \in t\_acceleration$
grd3 : $l\_loc \in t\_locations$

then

act1 : $v\_current := l\_speed$
act2 : $a\_current := l\_accel$
act3 : $loc\_current := l\_loc$

end

Event *new_outputs* $\widehat{=}$

any

l_ti_cmd
l_dmi_status

where

grd1 : $l\_ti\_cmd \in t\_TI\_commands$
grd2 : $l\_dmi\_status \in t\_DMI\_commands$

then

act1 : $cmd\_current := l\_ti\_cmd$
act2 : $status\_current := l\_dmi\_status$

end

END

4.3 Machine 1 - Brake Model

The first refinement adds the notion of the brake model. This is represented by the variables describing the speed dependent acceleration functions for emergency, service and normal service braking. The variables *T_brake_service* and *T_brake_emergency* describe the brake build-up times for the brakes.

MACHINE m1_brake_model
REFINES m0_trainstatus_commands
SEES c0_train_ti_dmi_commands
VARIABLES

A_brake_emergency emergency brake acceleration
A_brake_service service brake acceleration
A_brake_normal_service
T_brake_service
T_brake_emergency

EVENTS

Event *set_A_brake_emergency* $\widehat{=}$

any

l_a_brake

where

grd1 : $l\_a\_brake \in t\_speed \rightarrow t\_acceleration$

then

act1 : $A\_brake\_emergency := l\_a\_brake$

end

Event *set_A_brake_service* $\widehat{=}$

any
l_a_brake
where
grd1 : $l\_a\_brake \in t\_speed \to t\_acceleration$
then
act1 : $A\_brake\_service := l\_a\_brake$
end
Event $set\_A\_brake\_normal\_service \widehat{=}$
any
l_a_brake
where
grd1 : $l\_a\_brake \in t\_speed \to t\_acceleration$
then
act1 : $A\_brake\_normal\_service := l\_a\_brake$
end
Event $set\_T\_brake\_service \widehat{=}$
any
l_T_brake
where
grd1 : $l\_T\_brake \in t\_time$
then
act1 : $T\_brake\_service := l\_T\_brake$
end
Event $set\_T\_brake\_emergency \widehat{=}$
any
l_T_brake
where
grd1 : $l\_T\_brake \in t\_time$
then
act1 : $T\_brake\_emergency := l\_T\_brake$
end
END

4.4 Context 1 - Decelerations

This context extension adds a distance type and a function that maps the speed and distance to an acceleration.

CONTEXT c1_decelerations
EXTENDS c0_train_ti_dmi_commands
SETS
$t\_distance$
CONSTANTS
$f\_A\_deceleration0$
AXIOMS
axm1 : $f\_A\_deceleration0 \in t\_speed \times t\_distance \to t\_acceleration$
END

4.5 Machine 2 - Calculate Decelerations

This refinement adds the calculation of deceleration to the model. This is represented by three variables which are functions that map speed and distance to an acceleration. There is one function for each on of EBD, SBD and GUI.

MACHINE m2_calculation_deceleration
REFINES m1_brake_model
SEES c1_decelerations
VARIABLES

A_safe
$A\_expected$
$A\_normal\_service$

EVENTS
Event $set\_A\_safe \widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \to t\_acceleration$
grd2 : $A\_brake\_emergency \in t\_speed \to t\_acceleration$
then
act1 : $A\_safe := l\_a\_decel$
end
Event $set\_A\_expected \widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \to t\_acceleration$
grd2 : $A\_brake\_service \in t\_speed \to t\_acceleration$
then
act1 : $A\_expected := l\_a\_decel$
end
Event $set\_A\_normal\_service \widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \to t\_acceleration$
grd2 : $A\_brake\_normal\_service \in t\_speed \to t\_acceleration$
then
act1 : $A\_normal\_service := l\_a\_decel$
end
END

4.6 Machine 3 - Calculation of Brake Buildup Time

The next machine refinement adds the brake buildup calculation to the model. This is represented by two variables, T_be for the emergency brake and T_se for the service brake.

MACHINE m3_safe_brake_buildup
REFINES m2_calculation_deceleration
SEES c1_decelerations
VARIABLES

T_be

T_bs
EVENTS
Event $set\_T\_be \widehat{=}$
any
l_t_be
where
grd1 : $l\_t\_be \in t\_time$
grd2 : $T\_brake\_emergency \in t\_time$
then
act1 : $T\_be := l\_t\_be$
end
Event $set\_T\_bs \widehat{=}$
any
l_t_bs
where
grd1 : $l\_t\_bs \in t\_time$
grd2 : $T\_brake\_service \in t\_time$
then
act1 : $T\_bs := l\_t\_bs$
end
END

4.7 Machine 4 - Acceleration due to Gradient

The refinement adds the notion of the acceleration due to gradient. This is represented by the variable *A_gradient* which is a function that maps speed to acceleration.

MACHINE m4_accel_decel_gradient
REFINES m3_safe_brake_buildup
SEES c2_gradients
VARIABLES
$A\_gradient$
EVENTS
Event $set\_A\_gradient \widehat{=}$
any
$l\_a\_gradient$
where
grd1 : $l\_a\_gradient \in t\_acceleration$
then
act1 : $A\_gradient := l\_a\_gradient$
end
END

4.8 Context 3 - Speed Profiles

This context extension introduces the type *speed_profiles* which maps locations to speeds. It also defines one constant value of that type which is used as default value for variables of that type.

CONTEXT c3_speed_profiles
EXTENDS c2_gradients
CONSTANTS
$c\_speed\_profile0$

$t\_speed\_profiles$
AXIOMS

axm1 : $t\_speed\_profiles \subseteq t\_locations \times t\_speed$
axm2 : $c\_speed\_profile0 \in t\_speed\_profiles$
END

4.9 Machine 5 - Most Restrictive Speed Profile

This machine refinement introduces the most restrictive speed profile to the model. This is represented by the variable *MRSP* of the type *speed_profile*.

MACHINE m5_MRSP
REFINES m4_accel_decel_gradient
SEES c3_speed_profiles
VARIABLES

$MRSP$
EVENTS
Event $set\_MRSP \widehat{=}$
any
l_sp
where
grd1 : $l\_sp \in t\_speed\_profiles$
then
act1 : $MRSP := l\_sp$
end
END

4.10 Context 4 - Targets

This context extension introduces the type *t_targets* which represents a target, i.e., a pair of location and speed.

CONTEXT c4_targets
EXTENDS c3_speed_profiles
CONSTANTS

$t\_targets$
$c\_target0$
AXIOMS

axm1 : $t\_targets \subseteq t\_locations \times t\_speed$

axm2 : $c\_target0 \in t\_targets$
END

4.11 Machine 6 - Supervised Targets

This refinement adds limit of authority, end of authority and supervision limit to the model. For each there exists one variable of type *t_targets*.

MACHINE m6_supervised_targets
REFINES m5_MRSP

SEES c4_targets
VARIABLES

LOA
EOA
SvL
EVENTS
Event $set\_EOA \mathrel{\widehat{=}}$
any
l_target
where
grd1 : $l\_target \in t\_targets$
grd2 : $MRSP \in t\_speed\_profiles$
then
act1 : $EOA := l\_target$
end
Event $set\_LOA \mathrel{\widehat{=}}$
any
l_target
where
grd1 : $l\_target \in t\_targets$
grd2 : $MRSP \in t\_speed\_profiles$
then
act1 : $LOA := l\_target$
end
Event $set\_SvL \mathrel{\widehat{=}}$
any
l_target
where
grd1 : $l\_target \in t\_targets$
grd2 : $MRSP \in t\_speed\_profiles$
then
act1 : $SvL := l\_target$
end
END

4.12 Context 5 - Braking Curves

This context extension introduces the type $t\_braking\_curves$ and a constant of that type.

CONTEXT c5_braking_curves
EXTENDS c4_targets
SETS

$t\_braking\_curves$
CONSTANTS

c_curve0
AXIOMS

axm1 : $c\_curve0 \in t\_braking\_curves$
END

4.13 Machine 7 - Braking Curves

This machine refinement adds the braking curves to the model, these are represented by the three variables *EBD*, *SBD* and *GUI* of the appropriate type.

MACHINE m7_braking_curves
REFINES m6_supervised_targets
SEES c5_braking_curves
VARIABLES

EBD
SBD
GUI

EVENTS
Event $set\_EBD \;\widehat{=}$
any
l_curve
where
grd1 : $l\_curve \in t\_braking\_curves$
grd2 : $A\_safe \in t\_speed \times t\_distance \to t\_acceleration$
grd3 : $A\_expected \in t\_speed \times t\_distance \to t\_acceleration$
grd4 : $A\_normal\_service \in t\_speed \times t\_distance \to t\_acceleration$
grd5 : $LOA \in t\_targets$
grd6 : $EOA \in t\_targets$
grd7 : $SvL \in t\_targets$
then
act1 : $EBD := l\_curve$
end
Event $set\_SBD \;\widehat{=}$
any
l_curve
where
grd1 : $l\_curve \in t\_braking\_curves$
grd2 : $A\_safe \in t\_speed \times t\_distance \to t\_acceleration$
grd3 : $A\_expected \in t\_speed \times t\_distance \to t\_acceleration$
grd4 : $A\_normal\_service \in t\_speed \times t\_distance \to t\_acceleration$
grd5 : $LOA \in t\_targets$
grd6 : $EOA \in t\_targets$
grd7 : $SvL \in t\_targets$
then
act1 : $SBD := l\_curve$
end
Event $set\_GUI \;\widehat{=}$
any
l_curve
where
grd1 : $l\_curve \in t\_braking\_curves$
grd2 : $A\_safe \in t\_speed \times t\_distance \to t\_acceleration$
grd3 : $A\_expected \in t\_speed \times t\_distance \to t\_acceleration$
grd4 : $A\_normal\_service \in t\_speed \times t\_distance \to t\_acceleration$
grd5 : $LOA \in t\_targets$
grd6 : $EOA \in t\_targets$
grd7 : $SvL \in t\_targets$
then
act1 : $GUI := l\_curve$
end
END

4.14 Context 6 - Supervision Limits

This context adds the type *t_supervision_limits* to the model, as well as a constant value of that type.

CONTEXT c6_supervision_limits
EXTENDS c5_braking_curves
SETS

$t\_supervision\_limits$

CONSTANTS

$c\_slimit0$

AXIOMS

axm1 : $c\_slimit0 \in t\_supervision\_limits$

END

4.15 Machine 8 - Supervision Limit

This machine refinement adds the supervision limits to the model, emergency brake intervention (*EBI*), service brake intervention (*SBI*), warning limit (*warning_limit*), permitted speed (*P_limit*), indication limit (*I_limit*), pre-indication location (*PI_limit*) and the release start speed monitoring location (*RSM_start*).

MACHINE m8_supervision_limits
REFINES m7_braking_curves
SEES c6_supervision_limits
VARIABLES

EBI emergency brake intervention
SBI service brake intervention
W_limit warning limit
P_limit permitted speed
I_limit indication limit
PIl pre-indication_location
RSM_start release speed monitoring start location

EVENTS
Event $set\_EBI \mathrel{\widehat{=}}$

any

l_limit

where

grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$

then

act1 : $EBI := l\_limit$

end

Event $set\_SBI \mathrel{\widehat{=}}$

any

l_limit

where

grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$

grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$
then
act1 : $SBI := l\_limit$
end
Event $set\_W\_limit \mathrel{\widehat{=}}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$
then
act1 : $W\_limit := l\_limit$
end
Event $set\_I\_limit \mathrel{\widehat{=}}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$
then
act1 : $I\_limit := l\_limit$
end
Event $set\_P\_limit \mathrel{\widehat{=}}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$
then
act1 : $P\_limit := l\_limit$
end
Event $set\_PIl\_limit \mathrel{\widehat{=}}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$

grd7 : $T\_be \in t\_time$
then
act1 : $PIl := l\_limit$
end
Event $set\_RSM\_start\_limit \mathrel{\widehat{=}}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
grd2 : $MRSP \in t\_speed\_profiles$
grd3 : $EBD \in t\_braking\_curves$
grd4 : $SBD \in t\_braking\_curves$
grd5 : $GUI \in t\_braking\_curves$
grd6 : $T\_bs \in t\_time$
grd7 : $T\_be \in t\_time$
then
act1 : $RSM\_start := l\_limit$
end
END

5 Model Decomposition

The decomposition strategy chosen for this model is "A-style" (for Abrial) which means shared variable decomposition [SPHB11]. In this case the events of an Event-B machine are separated into n disjoint sets. Each of these sets represents one decomposed sub-machine. Variables can be shared between machines, if they are read / written by them. In this case-study, all events that write a single variable are in one machine, events that read this variable are in another machine. Only the sub-machine in which a variable is written can do data-refinement on these variables, in the other machines, these variables are marked as shared and cannot be refined.

For the model decomposition, an additional refinement of the model is necessary. The machine m9 does not really add detail to the refined machine m8. The only changes are that the variables which are read by an event are explicitly added to the conditions by specifying a typing condition for them. This assures that the model decomposition preserves these necessary variables in the sub-machines and only removes the unneeded ones.

The overview of the decomposition in the Rodin tool using the decomposition plug-in is shown in Fig. 4. It lists the decomposed machine, the decomposition style, the sub-components and the events in each sub-machine (only shown for a single one).

In the following sections, the sub-components, their respective refinement and the required context definitions will be explained.

5.1 dcmp - Braking Model

The braking model has the following input variables: *a_current*, *v_current* and *loc_current*. These variables are written by the event *update_train_state*. The variables and this event are marked as "DO NOT REFINE", i.e., they are shared variables and an external event.

The output variables are *T_brake_service*, *T_brake_emergency*, *A_brake_normal_service*, *A_brake_service* and *A_brake_emergency*.

MACHINE dcmp_braking_model_m0
SEES c6_supervision_limits
VARIABLES

A_brake_normal_service Private variable
A_brake_service Private variable
a_current Shared variable, DO NOT REFINE
v_current Shared variable, DO NOT REFINE

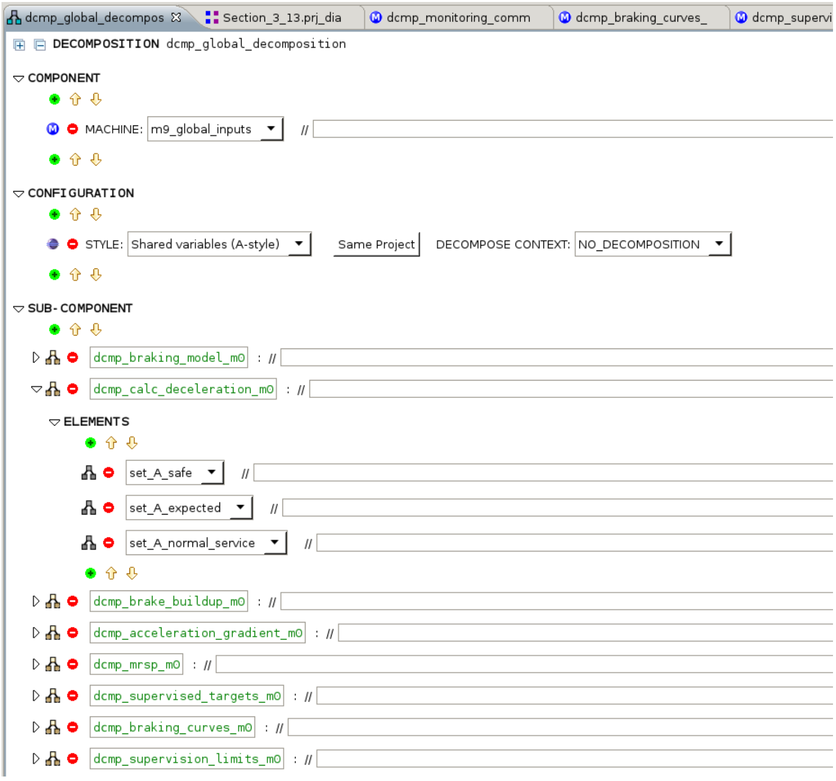


Figure 4. Decomposition Configuration

T_brake_service Shared variable, DO NOT REFINE
A_brake_emergency Private variable
loc_current Shared variable, DO NOT REFINE
T_brake_emergency Shared variable, DO NOT REFINE

EVENTS

Event *update_train_state* $\widehat{=}$

External event, DO NOT REFINE

any

l_speed
l_accel
l_loc

where

grd1 : $l\_speed \in t\_speed$
grd2 : $l\_accel \in t\_acceleration$
grd3 : $l\_loc \in t\_locations$

then

act1 : $v\_current := l\_speed$
act2 : $a\_current := l\_accel$
act3 : $loc\_current := l\_loc$

end

Event *set_A_brake_emergency* $\widehat{=}$

any

l_a_brake

where

grd1 : $l\_a\_brake \in t\_speed \rightarrow t\_acceleration$

then

act1 : $A\_brake\_emergency := l\_a\_brake$

end

Event *set_A_brake_service* $\widehat{=}$

any

l_a_brake
where
grd1 : $l\_a\_brake \in t\_speed \rightarrow t\_acceleration$
then
act1 : $A\_brake\_service := l\_a\_brake$
end
Event $set\_A\_brake\_normal\_service \widehat{=}$
any
l_a_brake
where
grd1 : $l\_a\_brake \in t\_speed \rightarrow t\_acceleration$
then
act1 : $A\_brake\_normal\_service := l\_a\_brake$
end
Event $set\_T\_brake\_service \widehat{=}$
any
l_T_brake
where
grd1 : $l\_T\_brake \in t\_time$
then
act1 : $T\_brake\_service := l\_T\_brake$
end
Event $set\_T\_brake\_emergency \widehat{=}$
any
l_T_brake
where
grd1 : $l\_T\_brake \in t\_time$
then
act1 : $T\_brake\_emergency := l\_T\_brake$
end
END

5.1.1 Context - Train Data 1 Brake

This context introduces the type *t_brake_percentage* and a function that takes the train speed, the brake percentage, the train length and the brake position as arguments and returns a Boolean value indicating whether the conversion model is applicable. It also introduces functions that calculate the conversion model for the emergency and for the service brake, as well as the conversion model for the brake buildup time of the service and emergency brake.

On the system level description of this Event-B model, these functions are not implemented in more detail. Their specification and implementation details can be found in §3.13.3 of the SRS.

CONTEXT ctx_train_data_1_brake
EXTENDS ctx_train_data
SETS
$t\_brake\_position$
CONSTANTS
$t\_brake\_percentage$
$c\_brake\_percentage0$
$f\_conversion\_model\_applicable$
$f\_conversion\_model\_A\_emergency$
$f\_conversion\_model\_A\_service$
$f\_conversion\_model\_T\_brake\_service$

f_conversion_model_T_brake_emergency

AXIOMS

axm1 : $t\_brake\_percentage \subseteq t\_percentage$

axm2 : $c\_brake\_percentage0 \in t\_brake\_percentage$

axm3 : $f\_conversion\_model\_applicable \in t\_speed \times t\_brake\_percentage \times t\_length \times t\_brake\_position \to BOOL$

cf. 3.13.3.2.1

axm4 : $f\_conversion\_model\_A\_emergency \in t\_brake\_percentage \to (t\_speed \to t\_acceleration)$

cf. 3.13.3.3

axm5 : $f\_conversion\_model\_A\_service \in t\_brake\_percentage \to (t\_speed \to t\_acceleration)$

axm6 : $f\_conversion\_model\_T\_brake\_service \in t\_brake\_position \times t\_length \times t\_speed \to t\_time$

cf. 3.13.3.4

axm7 : $f\_conversion\_model\_T\_brake\_emergency \in t\_brake\_position \times t\_length \times t\_speed \to t\_time$

END

5.1.2 dcmp - Braking Model Conversion Model

In the first refinement of the sub-machine adds the variables *brake_percentage* of type brake percentage and *brake_percentage_via_train_data* of Boolean type. These variables are modified by the event *specify_brake_percentage* and *remove_brake_percentage_data*. The Boolean variable signals the specification of the brake percentage via train data.

The value of the brake percentage is used in some events to compute concrete values for the conversion model instead of using an event parameter. For example in the event *calc_A_brake_emergency_conversion_model*, the function *f_conversion_model_A_emergency* is used with the value of*brake_percentage* to compute the conversion model. This replaces the event parameter *l_a_brake* by defining a *witness* for it.

MACHINE dcmp_braking_model_m1_conversion_model
REFINES dcmp_braking_model_m0
SEES c6_supervision_limits, ctx_train_data_1_brake
VARIABLES

brake_percentage

brake_percentage_via_train_data

EVENTS

Event *calc_A_brake_emergency_conversion_model* $\widehat{=}$
refines *set_A_brake_emergency*

any

l_brake_position

where

grd2 : $brake\_percentage\_via\_train\_data = TRUE$

grd3 : $f\_conversion\_model\_applicable(c\_train\_max\_speed \mapsto brake\_percentage \mapsto c\_train\_length \mapsto l\_brake\_position) = TRUE$

grd4 : $l\_brake\_position \in t\_brake\_position$

with

l_a_brake : $l\_a\_brake = f\_conversion\_model\_A\_emergency(brake\_percentage)$

then

act1 : $A\_brake\_emergency := f\_conversion\_model\_A\_emergency(brake\_percentage)$

end

Event *calc_A_brake_service_conversion_model* $\widehat{=}$
refines *set_A_brake_service*

any

l_brake_position

where

grd2 : $brake\_percentage\_via\_train\_data = TRUE$
grd3 : $f\_conversion\_model\_applicable(c\_train\_max\_speed \mapsto brake\_percentage \mapsto c\_train\_length \mapsto l\_brake\_position) = TRUE$
grd4 : $l\_brake\_position \in t\_brake\_position$

with

l_a_brake : $l\_a\_brake = f\_conversion\_model\_A\_service(brake\_percentage)$

then

act1 : $A\_brake\_service := f\_conversion\_model\_A\_service(brake\_percentage)$

end

Event *calc_T_brake_service_conversion_model* $\widehat{=}$
refines *set_T_brake_service*

any

$l\_brake\_position$
l_target_speed

where

grd2 : $brake\_percentage\_via\_train\_data = TRUE$
grd3 : $f\_conversion\_model\_applicable(c\_train\_max\_speed \mapsto brake\_percentage \mapsto c\_train\_length \mapsto l\_brake\_position) = TRUE$
grd4 : $l\_brake\_position \in t\_brake\_position$
grd5 : $l\_target\_speed \in t\_speed$

with

l_T_brake : $l\_T\_brake = f\_conversion\_model\_T\_brake\_service(l\_brake\_position \mapsto c\_train\_length \mapsto l\_target\_speed)$

then

act1 : $T\_brake\_service := f\_conversion\_model\_T\_brake\_service(l\_brake\_position \mapsto c\_train\_length \mapsto l\_target\_speed)$

end

refines *set_T_brake_emergency*

any

$l\_brake\_position$
l_target_speed

where

grd1 : $brake\_percentage\_via\_train\_data = TRUE$
grd2 : $f\_conversion\_model\_applicable(c\_train\_max\_speed \mapsto brake\_percentage \mapsto c\_train\_length \mapsto l\_brake\_position) = TRUE$
grd3 : $l\_brake\_position \in t\_brake\_position$
grd4 : $l\_target\_speed \in t\_speed$

with

l_T_brake : $l\_T\_brake = f\_conversion\_model\_T\_brake\_emergency(l\_brake\_position \mapsto c\_train\_length \mapsto l\_target\_speed)$

then

act1 : $T\_brake\_emergency := f\_conversion\_model\_T\_brake\_emergency(l\_brake\_position \mapsto c\_train\_length \mapsto l\_target\_speed)$

end

Event *specifiy_brake_percentage* $\widehat{=}$

any

l_brake

where

grd1 : $l\_brake \in t\_brake\_percentage$
grd2 : $brake\_percentage\_via\_train\_data = FALSE$

then

act1 : $brake\_percentage := l\_brake$
act2 : $brake\_percentage\_via\_train\_data := TRUE$

end

Event *remove_brake_percentage_data* $\widehat{=}$

when

grd1 : $brake\_percentage\_via\_train\_data = TRUE$

then

act1 : $brake\_percentage := c\_brake\_percentage0$
act2 : $brake\_percentage\_via\_train\_data := FALSE$

end

END

5.1.3 Context - Train Data 2 Direct Calculation

The next context introduces the type representing the combination of the different brake types of the train. It also defines functions that calculate the braking models for different brake combinations for the service brake and the emergency brake, as well as functions to calculate the brake buildup time for the service and emergency brake with the brake combination as argument. It also defines a function that calculates the normal brake function dependent on the brake position and the acceleration constants *c_SB01* and *c_SB02* (cf. §3.13.2.2.3.1.10).

CONTEXT ctx_train_data_2_dircect_calc
EXTENDS ctx_train_data_1_brake
SETS

$t\_brake\_combination$ cf. 3.13.2.2.3.1.7, i.e., combination of regenerative, eddy current and magnetic brake

CONSTANTS

c_v_zero target speed zero
$f\_calc\_A\_brake\_service\_direct$
$f\_calc\_A\_brake\_emergency\_direct$
$f\_calc\_A\_brake\_normal\_direct$ cf. 3.13.2.2.3.1.9
c_SB01
c_SB02
$f\_calc\_T\_brake\_service$
$f\_calc\_T\_brake\_emergency$

AXIOMS

axm1 : $c\_v\_zero \in t\_speed$
axm2 : $f\_calc\_A\_brake\_service\_direct \in t\_brake\_combination \to (t\_speed \to t\_acceleration)$
axm3 : $f\_calc\_A\_brake\_emergency\_direct \in t\_brake\_combination \to (t\_speed \to t\_acceleration)$
axm4 : $f\_calc\_A\_brake\_normal\_direct \in t\_brake\_position \times t\_acceleration \times t\_acceleration \to (t\_speed \to t\_acceleration)$

cf. 3.13.2.2.3.1.10
the accelerations are the c_SB01 and c_SB02 constants

axm5 : $c\_SB01 \in t\_acceleration$
axm6 : $c\_SB02 \in t\_acceleration$
axm7 : $f\_calc\_T\_brake\_service \in t\_brake\_combination \to t\_time$
axm8 : $f\_calc\_T\_brake\_emergency \in t\_brake\_combination \to t\_time$

END

5.1.4 dcmp - Braking Model Direct Calculation

This machine refinement uses the brake combinations and the corresponding functions from the context to produce more concrete events. For events that compute a function mapping speed to acceleration, the functions of the context and the constants for SB01 and SB02 are used to eliminate the event parameters by providing witnesses for them.

MACHINE dcmp_braking_model_m2_direct_calc
REFINES dcmp_braking_model_m1_conversion_model
SEES c6_supervision_limits, ctx_train_data_2_dircect_calc
EVENTS
Event $calc\_A\_brake\_emergency\_direct \;\widehat{=}$
refines $set\_A\_brake\_emergency$

any

$l\_brake\_combination$

where

`grd1` : $l\_brake\_combination \in t\_brake\_combination$

with

`l_a_brake` : $l\_a\_brake = f\_calc\_A\_brake\_emergency\_direct(l\_brake\_combination)$

then

`act1` : $A\_brake\_emergency := f\_calc\_A\_brake\_emergency\_direct(l\_brake\_combination)$

end

Event $calc\_A\_brake\_service\_direct \mathrel{\widehat{=}}$
refines $set\_A\_brake\_service$

any

$l\_brake\_combination$

where

`grd2` : $l\_brake\_combination \in t\_brake\_combination$

with

`l_a_brake` : $l\_a\_brake = f\_calc\_A\_brake\_service\_direct(l\_brake\_combination)$

then

`act1` : $A\_brake\_service := f\_calc\_A\_brake\_service\_direct(l\_brake\_combination)$

end

Event $set\_A\_brake\_normal\_service \mathrel{\widehat{=}}$
refines $set\_A\_brake\_normal\_service$

any

$l\_brake\_position$

where

`grd1` : $l\_brake\_position \in t\_brake\_position$

with

`l_a_brake` : $l\_a\_brake = f\_calc\_A\_brake\_normal\_direct(l\_brake\_position \mapsto c\_SB01 \mapsto c\_SB02)$

then

`act1` : $A\_brake\_normal\_service := f\_calc\_A\_brake\_normal\_direct(l\_brake\_position \mapsto c\_SB01 \mapsto c\_SB02)$

end

Event $set\_T\_brake\_service \mathrel{\widehat{=}}$
refines $set\_T\_brake\_service$

any

$l\_brake\_combination$

where

`grd1` : $l\_brake\_combination \in t\_brake\_combination$

with

`l_T_brake` : $l\_T\_brake = f\_calc\_T\_brake\_service(l\_brake\_combination)$

then

`act1` : $T\_brake\_service := f\_calc\_T\_brake\_service(l\_brake\_combination)$

end

Event $set\_T\_brake\_emergency \mathrel{\widehat{=}}$
refines $set\_T\_brake\_emergency$

any

$l\_brake\_combination$

where

`grd1` : $l\_brake\_combination \in t\_brake\_combination$

with

`l_T_brake` : $l\_T\_brake = f\_calc\_T\_brake\_emergency(l\_brake\_combination)$

then

`act1` : $T\_brake\_emergency := f\_calc\_T\_brake\_emergency(l\_brake\_combination)$

end

END

5.2 dcmp - Calc Deceleration

The deceleration calculation model has the following input variables: *a_current*, *v_current* and *loc_current*. The output variables it calculates are *A_normal_service*, *A_expected* and *A_safe* which each map train speed and distance to an acceleration.

MACHINE dcmp_calc_deceleration_m0
SEES c6_supervision_limits
EVENTS
Event *set_A_safe* $\widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$
then
act1 : $A\_safe := l\_a\_decel$
end
Event *set_A_expected* $\widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$
then
act1 : $A\_expected := l\_a\_decel$
end
Event *set_A_normal_service* $\widehat{=}$
any
l_a_decel
where
grd1 : $l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$
then
act1 : $A\_normal\_service := l\_a\_decel$
end
END

5.3 dcmp - Brake Buildup

The brake buildup time calculation has the following input variables: *a_current*, *v_current* and *loc_current*. The output variables are *T_be*, *T_se*, *T_brake_service* and *T_brake_emergency*.

MACHINE dcmp_brake_buildup_m0
SEES c6_supervision_limits
VARIABLES

T_be Shared variable, DO NOT REFINE
T_bs Shared variable, DO NOT REFINE
a_current Shared variable, DO NOT REFINE
v_current Shared variable, DO NOT REFINE
T_brake_service Shared variable, DO NOT REFINE
loc_current Shared variable, DO NOT REFINE
T_brake_emergency Shared variable, DO NOT REFINE

INVARIANTS

typing_T_be : $T\_be \in t\_time$
typing_T_bs : $T\_bs \in t\_time$

`typing_a_current` : $a\_current \in t\_acceleration$
`typing_v_current` : $v\_current \in t\_speed$
`typing_T_brake_service` : $T\_brake\_service \in t\_time$
`typing_loc_current` : $loc\_current \in t\_locations$
`typing_T_brake_emergency` : $T\_brake\_emergency \in t\_time$

EVENTS

Initialisation

begin

`act1` : $v\_current := c\_v0$
`act2` : $a\_current := c\_a0$
`act3` : $loc\_current := c\_l0$
`act9` : $T\_brake\_service := c\_T\_brake0$
`act10` : $T\_brake\_emergency := c\_T\_brake0$
`act14` : $T\_be := c\_T\_brake0$
`act15` : $T\_bs := c\_T\_brake0$

end

Event $update\_train\_state \widehat{=}$

External event, DO NOT REFINE

any

l_speed
l_accel
l_loc

where

`grd1` : $l\_speed \in t\_speed$
`grd2` : $l\_accel \in t\_acceleration$
`grd3` : $l\_loc \in t\_locations$

then

`act1` : $v\_current := l\_speed$
`act2` : $a\_current := l\_accel$
`act3` : $loc\_current := l\_loc$

end

Event $set\_T\_brake\_service \widehat{=}$

External event, DO NOT REFINE

any

l_T_brake

where

`grd1` : $l\_T\_brake \in t\_time$

then

`act1` : $T\_brake\_service := l\_T\_brake$

end

Event $set\_T\_brake\_emergency \widehat{=}$

External event, DO NOT REFINE

any

l_T_brake

where

`grd1` : $l\_T\_brake \in t\_time$

then

`act1` : $T\_brake\_emergency := l\_T\_brake$

end

Event $set\_T\_be \widehat{=}$

any

l_t_be

where

`grd1` : $l\_t\_be \in t\_time$

then

`act1` : $T\_be := l\_t\_be$

end
Event $set\_T\_bs \widehat{=}$
any
l_t_bs
where
grd1 : $l\_t\_bs \in t\_time$
then
act1 : $T\_bs := l\_t\_bs$
end
END

5.3.1 Context - Brake Buildup Functions

This context defines 4 functions that calculate the brake buildup time for the service and emergency brake. For each there are two functions, one for the conversion model and one without conversion model. The service break is not safety-critical, therefore it does not take a correction factor into account (cf. §3.13.6.3.1.1).

CONTEXT ctx_functions_1_calc_T_brake
EXTENDS ctx_functions_0
CONSTANTS
$f\_T\_brake\_service\_conversion$
$f\_T\_brake\_service$
$f\_T\_brake\_emergency\_conversion$
$f\_T\_brake\_emergency$
AXIOMS
axm1 : $f\_T\_brake\_service\_conversion \in t\_time \times t\_brake\_combination \rightarrow t\_time$
axm2 : $f\_T\_brake\_service \in t\_time \times t\_brake\_combination \rightarrow t\_time$
axm3 : $f\_T\_brake\_emergency\_conversion \in t\_time \times t\_correction\_factor \rightarrow t\_time$
axm4 : $f\_T\_brake\_emergency \in t\_time \times t\_brake\_combination \rightarrow t\_time$
END

5.3.2 dcmp - Brake Buildup Calculation

This refinement applies the calculation functions defined in the context to the calculation events. It adds the notions of train data to the machine which controls whether the conversion model is used for the brake buildup time calculation. This is represented by two variables each, a Boolean variable that signals whether a concrete train data has been supplied and a variable of type *t_time* representing the supplied value.

MACHINE dcmp_brake_buildup_m2_calculation
REFINES dcmp_brake_buildup_m0
SEES c6_supervision_limits, ctx_functions_1_calc_T_brake
VARIABLES
$T\_brake\_service\_data$
$T\_brake\_service\_via\_train\_data$
$T\_brake\_emergency\_data$
$T\_brake\_emergency\_via\_train\_data$
INVARIANTS
inv1 : $T\_brake\_service\_data \in t\_time$
inv2 : $T\_brake\_service\_via\_train\_data \in BOOL$
inv3 : $T\_brake\_emergency\_data \in t\_time$
inv4 : $T\_brake\_emergency\_via\_train\_data \in BOOL$
EVENTS

Event $set\_T\_be \widehat{=}$
refines set_T_be

any

$l\_brake\_combination$

where

grd1 : $T\_brake\_emergency\_via\_train\_data = TRUE$
grd2 : $l\_brake\_combination \in t\_brake\_combination$

with

l_t_be : $l\_t\_be = f\_T\_brake\_emergency(T\_brake\_emergency \mapsto l\_brake\_combination)$

then

act1 : $T\_be := f\_T\_brake\_emergency(T\_brake\_emergency \mapsto l\_brake\_combination)$

end

Event $calc\_T\_be\_conversion\_model \widehat{=}$
refines $set\_T\_brake\_emergency$

when

grd1 : $T\_brake\_emergency\_via\_train\_data = FALSE$

with

l_T_brake : $l\_T\_brake = f\_T\_brake\_emergency\_conversion(T\_brake\_service \mapsto c\_Kt\_int)$

then

act1 : $T\_brake\_emergency := f\_T\_brake\_emergency\_conversion(T\_brake\_service \mapsto c\_Kt\_int)$

end

Event $set\_T\_bs \widehat{=}$
refines set_T_bs

any

$l\_brake\_combination$

where

grd1 : $T\_brake\_service\_via\_train\_data = TRUE$
grd2 : $l\_brake\_combination \in t\_brake\_combination$

with

l_t_bs : $l\_t\_bs = f\_T\_brake\_service(T\_brake\_service\_data \mapsto l\_brake\_combination)$

then

act1 : $T\_bs := f\_T\_brake\_service(T\_brake\_service\_data \mapsto l\_brake\_combination)$

end

Event $calc\_T\_bs\_conversion\_model \widehat{=}$
refines set_T_bs

any

$l\_brake\_combination$

where

grd1 : $T\_brake\_service\_via\_train\_data = FALSE$
grd2 : $l\_brake\_combination \in t\_brake\_combination$

with

l_t_bs : $l\_t\_bs = f\_T\_brake\_emergency(T\_brake\_service \mapsto l\_brake\_combination)$

then

act1 : $T\_bs := f\_T\_brake\_emergency(T\_brake\_service \mapsto l\_brake\_combination)$

end

Event $specify\_T\_brake\_service\_train\_data \widehat{=}$

any

l_T_brake

where

grd1 : $l\_T\_brake \in t\_time$
grd2 : $T\_brake\_service\_via\_train\_data = FALSE$

then

act1 : $T\_brake\_service\_data := l\_T\_brake$
act2 : $T\_brake\_service\_via\_train\_data := TRUE$

end
Event *remove_T_brake_service_data* $\widehat{=}$
when
grd1 : $T\_brake\_service\_via\_train\_data = TRUE$
then
act1 : $T\_brake\_service\_via\_train\_data := FALSE$
end
Event *specify_T_brake_emergency_train_data* $\widehat{=}$
any
l_T_brake
where
grd1 : $l\_T\_brake \in t\_time$
grd2 : $T\_brake\_emergency\_via\_train\_data = FALSE$
then
act1 : $T\_brake\_emergency\_data := l\_T\_brake$
act2 : $T\_brake\_emergency\_via\_train\_data := TRUE$
end
Event *remove_T_brake_emergency_data* $\widehat{=}$
when
grd1 : $T\_brake\_emergency\_via\_train\_data = TRUE$
then
act1 : $T\_brake\_emergency\_via\_train\_data := FALSE$
end
END

5.4 dcmp - Acceleration Gradient

The calculation of the acceleration gradient has the current train state as input values (i.e., speed, location and acceleration). Its output value is the acceleration due to gradient, represented by the variable *A_gradient*.

MACHINE dcmp_acceleration_gradient_m0
SEES c6_supervision_limits
VARIABLES

a_current Shared variable, DO NOT REFINE
v_current Shared variable, DO NOT REFINE
loc_current Shared variable, DO NOT REFINE
A_gradient Private variable

INVARIANTS

typing_a_current : $a\_current \in t\_acceleration$
typing_v_current : $v\_current \in t\_speed$
typing_loc_current : $loc\_current \in t\_locations$
typing_A_gradient : $A\_gradient \in t\_acceleration$
m4_accel_decel_gradient_inv1 : $A\_gradient \in t\_acceleration$

EVENTS
Initialisation
begin
act1 : $v\_current := c\_v0$
act2 : $a\_current := c\_a0$
act3 : $loc\_current := c\_l0$
act16 : $A\_gradient := c\_a0$
end
Event *update_train_state* $\widehat{=}$
External event, DO NOT REFINE
any

l_speed
l_accel
l_loc

where

grd1 : $l\_speed \in t\_speed$
grd2 : $l\_accel \in t\_acceleration$
grd3 : $l\_loc \in t\_locations$

then

act1 : $v\_current := l\_speed$
act2 : $a\_current := l\_accel$
act3 : $loc\_current := l\_loc$

end

Event $set\_A\_gradient \;\widehat{=}$

any

$l\_a\_gradient$

where

grd1 : $l\_a\_gradient \in t\_acceleration$

then

act1 : $A\_gradient := l\_a\_gradient$

end

END

5.4.1 Context - MRSP and Gradients

This context introduces functions to calculate an acceleration due to gradient (cf. 3.13.4) and to calculate the current most restrictive speed profile from the set of applicable speed profiles for a location.

CONTEXT ctx_functions_0
EXTENDS ctx_type_gradients
CONSTANTS

$f\_accel\_due\_gradient$
f_grad_uphill
$f\_compensate\_gradient\_profile$
f_mrsp

AXIOMS

axm1 : $f\_accel\_due\_gradient \in t\_locations \times t\_locations \times t\_gradients \times t\_mass\_percentage \times t\_speed\_profiles \to t\_acceleration$

(loc, SvL, compensated_grad, mass, speed_profile)
calculates acceleration due to gradient according to 3.13.4

axm2 : $f\_grad\_uphill \in t\_gradients \to BOOL$

indicates whether gradient is uphil or downhill

axm3 : $f\_compensate\_gradient\_profile \in t\_locations \times t\_locations \times t\_gradient\_profiles \to t\_gradients$

(loc, SvL, profile) compensates for gradient profile (cf. 3.13.4.2.1)

axm4 : $f\_mrsp \in t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_speed\_profiles \times t\_train\_modes \times t\_speed \times t\_length \to t\_speed\_profiles$

function which calculates the most restrictive speed profile
(MRSP) from the following inputs:
SSP, Axle Load SP, TSR, signalling related, mode related, LX speed,
override function related, SR to ensure permitted brake distance,
train mode, max train speed, train length

END

5.4.2 dcmp - Rotating Mass

This machine introduces the notion of rotating mass into the model. the rotating mass is represented by two variables, the Boolean one *rotating_mass_specified* which signals whether an explicit value for the mass has been specified and *M_rotating_mass* which represents a concrete value, given as percentage of the train mass.

Here the event *set_A_gradient* the event parameter *l_a_gradient* is replaced by a witness which is the application of the function to calculate the gradient to the train location and the event parameters.

MACHINE dcmp_acceleration_gradient_m1
REFINES dcmp_acceleration_gradient_m0
SEES c6_supervision_limits, ctx_functions_0
VARIABLES

$rotating\_mass\_specified$
$M\_rotating\_nom$

INVARIANTS

inv1 : $rotating\_mass\_specified \in BOOL$
inv2 : $M\_rotating\_nom \in t\_mass\_percentage$

EVENTS
Event $set\_A\_gradient \widehat{=}$
refines $set\_A\_gradient$

any

l_mass
l_SvL
l_grad
l_TSR

where

grd2 : $l\_mass \in t\_mass\_percentage$
grd3 : $l\_SvL \in t\_locations$
grd4 : $l\_grad \in t\_gradients$
grd5 : $l\_TSR \in t\_speed\_profiles$

with

l_a_gradient : $l\_a\_gradient = f\_accel\_due\_gradient(loc\_current \mapsto l\_SvL \mapsto l\_grad \mapsto l\_mass \mapsto l\_TSR)$

then

act1 : $A\_gradient := f\_accel\_due\_gradient(loc\_current \mapsto l\_SvL \mapsto l\_grad \mapsto l\_mass \mapsto l\_TSR)$

end

Event $specify\_rotating\_mass \widehat{=}$

any

l_mass

where

grd1 : $rotating\_mass\_specified = FALSE$
grd2 : $l\_mass \in t\_mass\_percentage$

then

act1 : $M\_rotating\_nom := l\_mass$
act2 : $rotating\_mass\_specified := TRUE$

end

Event $delete\_rotating\_mass\_data \widehat{=}$

when

grd1 : $rotating\_mass\_specified = TRUE$

then

act1 : $rotating\_mass\_specified := FALSE$
act2 : $M\_rotating\_nom := c\_mass\_percentage0$

end

END

5.4.3 dcmp - Acceleration due to Gradient Calculation

This machine refines the calculation of the acceleration due to gradient by splitting the calculation event into three refined ones. One to calculate the acceleration if a concrete rotating mass has been specified and two for an unknown rotating mass. In this case, there is one event for an uphill gradient which uses the maximal rotating mass value (constant *c_M_rotating_max* and for a downhill gradient which uses the minimal rotating mass (c_M_rotating_min) for the calculation (cf. 3.13.4.3.2).

MACHINE dcmp_acceleration_gradient_m2_grad_accel
REFINES dcmp_acceleration_gradient_m1
SEES c6_supervision_limits, ctx_functions_0
EVENTS
Event *calc_A_gradient_mass_known* $\widehat{=}$
refines *set_A_gradient*

any

l_SvL
l_grad
l_TSR

where

grd3 : $l\_SvL \in t\_locations$
grd4 : $l\_grad \in t\_gradients$
grd5 : $l\_TSR \in t\_speed\_profiles$
grd6 : $rotating\_mass\_specified = TRUE$

with

l_mass : $l\_mass = M\_rotating\_nom$

then

act1 : $A\_gradient := f\_accel\_due\_gradient(loc\_current \mapsto l\_SvL \mapsto l\_grad \mapsto M\_rotating\_nom \mapsto l\_TSR)$

end

Event *calc_A_gradient_mass_unknown_uphill* $\widehat{=}$
refines *set_A_gradient*

any

l_SvL
l_grad
l_TSR

where

grd3 : $l\_SvL \in t\_locations$
grd4 : $l\_grad \in t\_gradients$
grd5 : $l\_TSR \in t\_speed\_profiles$
grd6 : $f\_grad\_uphill(l\_grad) = TRUE$
grd7 : $rotating\_mass\_specified = FALSE$

with

l_mass : $l\_mass = c\_M\_rotating\_max$

then

act1 : $A\_gradient := f\_accel\_due\_gradient(loc\_current \mapsto l\_SvL \mapsto l\_grad \mapsto c\_M\_rotating\_max \mapsto l\_TSR)$

end

Event *calc_A_gradient_mass_unknown_downhill* $\widehat{=}$
refines *set_A_gradient*

any

l_SvL
l_grad
l_TSR

where

grd3 : $l\_SvL \in t\_locations$
grd4 : $l\_grad \in t\_gradients$
grd5 : $l\_TSR \in t\_speed\_profiles$
grd6 : $f\_grad\_uphill(l\_grad) = FALSE$
grd7 : $rotating\_mass\_specified = FALSE$

with

`l_mass` : $l\_mass = c\_M\_rotating\_min$

then

`act1` : $A\_gradient := f\_accel\_due\_gradient(loc\_current \mapsto l\_SvL \mapsto l\_grad \mapsto c\_M\_rotating\_min \mapsto l\_TSR)$

end

END

5.5 dcmp - MRSP

This machine introduces the notion of the most restrictive speed profile (MRPS) into the system. The input for this machine is the current train state, the variable *MRSP* of type *speed_profile* is its output value.

MACHINE dcmp_mrsp_m0
SEES c6_supervision_limits
VARIABLES

$MRSP$ Shared variable, DO NOT REFINE
$a\_current$ Shared variable, DO NOT REFINE
$v\_current$ Shared variable, DO NOT REFINE
$loc\_current$ Shared variable, DO NOT REFINE

INVARIANTS

`typing_MRSP` : $MRSP \in t\_speed\_profiles$
`typing_a_current` : $a\_current \in t\_acceleration$
`typing_v_current` : $v\_current \in t\_speed$
`typing_loc_current` : $loc\_current \in t\_locations$

EVENTS

Initialisation

begin

`act1` : $v\_current := c\_v0$
`act2` : $a\_current := c\_a0$
`act3` : $loc\_current := c\_l0$
`act17` : $MRSP := c\_speed\_profile0$

end

Event $update\_train\_state \mathrel{\widehat=}$

External event, DO NOT REFINE

any

l_speed
l_accel
l_loc

where

`grd1` : $l\_speed \in t\_speed$
`grd2` : $l\_accel \in t\_acceleration$
`grd3` : $l\_loc \in t\_locations$

then

`act1` : $v\_current := l\_speed$
`act2` : $a\_current := l\_accel$
`act3` : $loc\_current := l\_loc$

end

Event $set\_MRSP \mathrel{\widehat=}$

any

l_sp

where

`grd1` : $l\_sp \in t\_speed\_profiles$

then

`act1` : $MRSP := l\_sp$

end

END

5.5.1 dcmp - All Speed Profiles

This machine refines the calculation of the MRSP by refining the event *set_MRSP* to an event that takes all possible speed restrictions as event parameters and uses the function of the context of Section 5.4.1 to calculate the MRSP (cf. 3.11).

MACHINE dcmp_mrsp_m1_all_speed_profiles
REFINES dcmp_mrsp_m0
SEES c6_supervision_limits, ctx_functions_0
EVENTS
Event *calculate_MRSP_from_all_speed_restrictions* $\widehat{=}$
refines *set_MRSP*

any

l_SSP
$l\_AxleLoadSP$
l_TSR
$l\_SignalRelatedSP$
l_mode
$l\_modeRelatedSP$
l_LXSP
$l\_OverrideFuncSP$
$l\_ensureBrakingDistanceSP$

where

grd2 : $l\_SSP \in t\_speed\_profiles$
grd3 : $l\_AxleLoadSP \in t\_speed\_profiles$
grd4 : $l\_TSR \in t\_speed\_profiles$
grd5 : $l\_SignalRelatedSP \in t\_speed\_profiles$
grd6 : $l\_mode \in t\_train\_modes$
grd7 : $l\_modeRelatedSP \in t\_speed\_profiles$
grd8 : $l\_LXSP \in t\_speed\_profiles$
grd9 : $l\_OverrideFuncSP \in t\_speed\_profiles$
grd10 : $l\_ensureBrakingDistanceSP \in t\_speed\_profiles$

with

l_sp : $l\_sp = f\_mrsp(l\_SSP \mapsto l\_AxleLoadSP \mapsto l\_TSR \mapsto l\_SignalRelatedSP \mapsto l\_modeRelatedSP \mapsto l\_LXSP \mapsto l\_OverrideFuncSP \mapsto l\_ensureBrakingDistanceSP \mapsto l\_mode \mapsto c\_train\_max\_speed \mapsto c\_train\_length)$

then

act1 : $MRSP := f\_mrsp(l\_SSP \mapsto l\_AxleLoadSP \mapsto l\_TSR \mapsto l\_SignalRelatedSP \mapsto l\_modeRelatedSP \mapsto l\_LXSP \mapsto l\_OverrideFuncSP \mapsto l\_ensureBrakingDistanceSP \mapsto l\_mode \mapsto c\_train\_max\_speed \mapsto c\_train\_length)$

end

END

5.6 dcmp - Supervised Targets

This machine introduces has the current train state as inputs and produces as outputs the supervised locations: end of authority, limit of authority and supervised location, represented by *EOA*, *LOA* and *SvL*, each of type *t_targets*.

MACHINE dcmp_supervised_targets_m0
SEES c6_supervision_limits
VARIABLES

SvL Shared variable, DO NOT REFINE
LOA Shared variable, DO NOT REFINE
EOA Shared variable, DO NOT REFINE

EVENTS

Event *set_EOA* $\widehat{=}$
  any
    l_target
  where
    `grd1` : $l\_target \in t\_targets$
  then
    `act1` : $EOA := l\_target$
  end
Event *set_LOA* $\widehat{=}$
  any
    l_target
  where
    `grd1` : $l\_target \in t\_targets$
  then
    `act1` : $LOA := l\_target$
  end
Event *set_SvL* $\widehat{=}$
  any
    l_target
  where
    `grd1` : $l\_target \in t\_targets$
  then
    `act1` : $SvL := l\_target$
  end
END

5.7 dcmp - Braking Curves

This machine takes as input the current train state, the computed accelerations and the supervised targets. Its outputs are the braking curves for the mergency brake (*EBD*), the service brake (*SBD*) and the graphical representation for the driver (*GUI*).

MACHINE dcmp_braking_curves_m0
SEES c6_supervision_limits
VARIABLES
  SBD Shared variable, DO NOT REFINE
  GUI Shared variable, DO NOT REFINE
  EBD Shared variable, DO NOT REFINE
INVARIANTS
  `typing_SBD` : $SBD \in t\_braking\_curves$
  `typing_GUI` : $GUI \in t\_braking\_curves$
  `typing_EBD` : $EBD \in t\_braking\_curves$
EVENTS
Event *set_EBD* $\widehat{=}$
  any
    l_curve
  where
    `grd1` : $l\_curve \in t\_braking\_curves$
  then
    `act1` : $EBD := l\_curve$
  end
Event *set_SBD* $\widehat{=}$
  any

l_curve
where
grd1 : $l\_curve \in t\_braking\_curves$
then
act1 : $SBD := l\_curve$
end
Event *set_GUI* $\widehat{=}$
any
l_curve
where
grd1 : $l\_curve \in t\_braking\_curves$
then
act1 : $GUI := l\_curve$
end
END

5.8 dcmp - Supervision Limits

This machine takes as inputs the current train state, the braking curves, the MRSP and the brake buildup times. It produces as outputs the following supervision limits: emergency brake intervention (*EBI*), service brake intervention (*SBI*), warning limit (*W_limit*), permitted (*P_limit*), indication limit (*I_limit*), pre-indication location (*PIl*) and the release speed monitoring start location (*RSM_start*). Each of these variables is of type *t_supervision_limits*.

MACHINE dcmp_supervision_limits_m0
SEES c6_supervision_limits
VARIABLES
SBI Shared variable, DO NOT REFINE
PIl Shared variable, DO NOT REFINE
W_limit Shared variable, DO NOT REFINE
RSM_start Shared variable, DO NOT REFINE
P_limit Shared variable, DO NOT REFINE
EBI Shared variable, DO NOT REFINE
INVARIANTS
typing_SBI : $SBI \in t\_supervision\_limits$
typing_I_limit : $I\_limit \in t\_supervision\_limits$
typing_PIl : $PIl \in t\_supervision\_limits$
typing_W_limit : $W\_limit \in t\_supervision\_limits$
typing_RSM_start : $RSM\_start \in t\_supervision\_limits$
typing_P_limit : $P\_limit \in t\_supervision\_limits$
typing_EBI : $EBI \in t\_supervision\_limits$
EVENTS
Event *set_EBI* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $EBI := l\_limit$
end
Event *set_SBI* $\widehat{=}$
any
l_limit
where

grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $SBI := l\_limit$
end
Event *set_W_limit* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $W\_limit := l\_limit$
end
Event *set_I_limit* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $I\_limit := l\_limit$
end
Event *set_P_limit* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $P\_limit := l\_limit$
end
Event *set_PIl_limit* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $PIl := l\_limit$
end
Event *set_RSM_start_limit* $\widehat{=}$
any
l_limit
where
grd1 : $l\_limit \in t\_supervision\_limits$
then
act1 : $RSM\_start := l\_limit$
end
END

5.9 dcmp - Monitoring Commands

This machine takes as inputs the SBI, indication limit, pre-indication limit, warning limit, release speed monitoring start limit, permitted speed limit and the EBI. Its outputs are the DMI and the TI commands, i.e., the variables *status_current* of type *t_DMI_commands* and *cmd_current* of type *t_TI_commands*.

MACHINE dcmp_monitoring_commands

SEES c6_supervision_limits
VARIABLES

status_current Private variable
cmd_current Private variable

INVARIANTS

`typing_status_current` : $status\_current \in t\_DMI\_commands$
`typing_cmd_current` : $cmd\_current \in t\_TI\_commands$

EVENTS
Event *new_outputs* $\widehat{=}$
any

l_ti_cmd
l_dmi_status

where

`grd1` : $l\_ti\_cmd \in t\_TI\_commands$
`grd2` : $l\_dmi\_status \in t\_DMI\_commands$

then

`act1` : $cmd\_current := l\_ti\_cmd$
`act2` : $status\_current := l\_dmi\_status$

end
END

References

[Abr10] Jean-Raymond Abrial. *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, 2010.

[Eur12] European Railway Agency (ERA). System Requirements Specification - ETCS Subset 026. http://www.era.europa.eu/Document-Register/Documents/Index00426.zip, 2012.

[Jas12] Michael Jastram, editor. *Rodin User's Handbook*. DEPLOY Project, 2012.

[Mat13a] Matthias Güdemann. Event-B Model of Subset 026, Section 3.5. github - openETCS, 2013.

[Mat13b] Matthias Güdemann. Event-B Model of Subset 026, Section 4.6. github - openETCS, 2013.

[Mat13c] Matthias Güdemann. Event-B Model of Subset 026, Section 5.9. github - openETCS, 2013.

[SPHB11] Renato Silva, Carine Pascal, Thai Son Hoang, and Michael Butler. Decomposition tool for event-B. *Software: Practice and Experience*, 41(2):199–208, 2011.