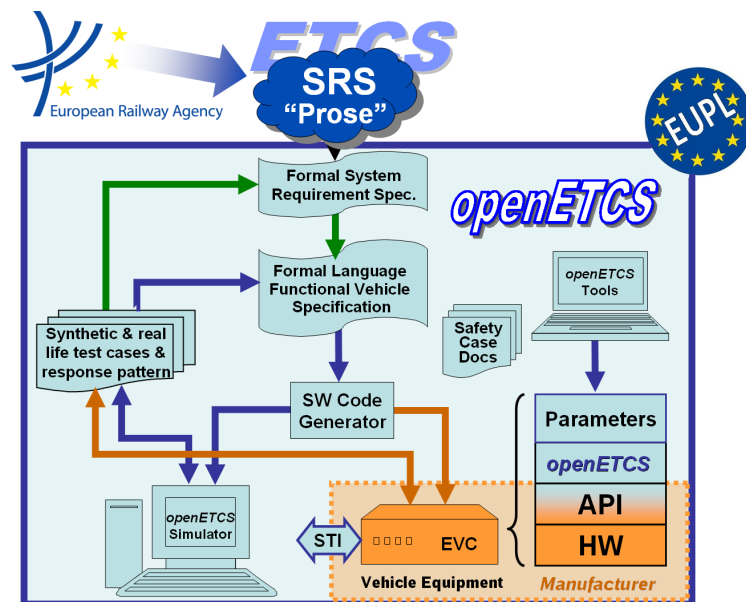


Work-Package 7: "Toolchain"

## Event-B Model of Subset 026, Section 3.13

Matthias Güdemann

June 2013



Funded by:



This page is intentionally left blank

**Work-Package 7: “Toolchain”**

**OETCS  
June 2013**

## Event-B Model of Subset 026, Section 3.13

Matthias Güdemann

Systemel  
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

**Disclaimer:** This work is licensed under the "openETCS Open License Terms" (oOLT) dual Licensing: European Union Public Licence (EURL v.1.1+) AND Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

THE WORK IS PROVIDED UNDER openETCS OPEN LICENSE TERMS (oOLT) WHICH IS A DUAL LICENSE AGREEMENT INCLUDING THE TERMS OF THE EUROPEAN UNION PUBLIC LICENSE (VERSION 1.1 OR ANY LATER VERSION) AND THE TERMS OF THE CREATIVE COMMONS PUBLIC LICENSE ("CCPL"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS OLT LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

<http://creativecommons.org/licenses/by-sa/3.0/>

<http://joinup.ec.europa.eu/software/page/eupl/licence-eupl>

# 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 .....	12
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 .....	16
4.14	Context 6 - Supervision Limits .....	17
4.15	Machine 8 - Supervision Limit.....	17
5	Model Decomposition.....	19
5.1	dcmp - Braking Model.....	19
5.2	dcmp - Calc Deceleration .....	26
5.3	dcmp - Brake Buildup .....	27
5.4	dcmp - Acceleration Gradient .....	30
5.5	dcmp - MRSP .....	35
5.6	dcmp - Supervised Targets .....	36
5.7	dcmp - Braking Curves .....	37
5.8	dcmp - Supervision Limits.....	38
5.9	dcmp - Monitoring Commands.....	40
	References .....	40

# 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..... 20

**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-Systemrel/Event\\_B/rodin-projects-github.pdf?raw=true](https://github.com/openETCS/model-evaluation/blob/master/model/B-Systemrel/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 them and these values are then 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

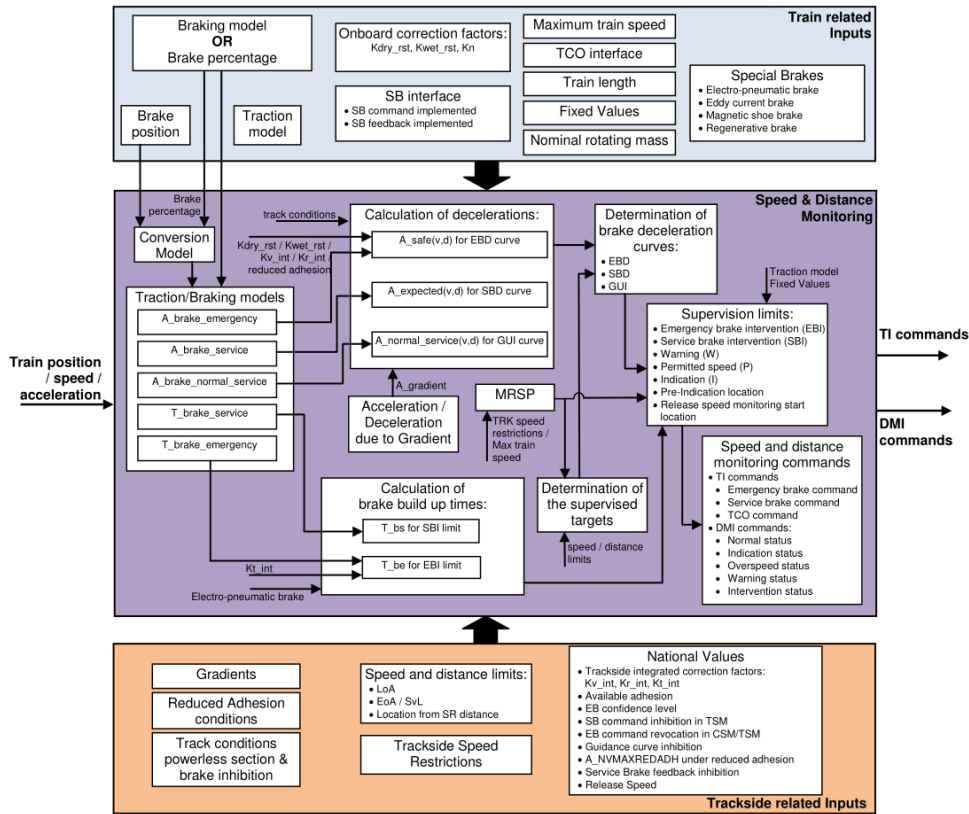


Figure 1. Speed and Distance Monitoring Overview ([Eur12] p. 85)

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<sup>1</sup> 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.

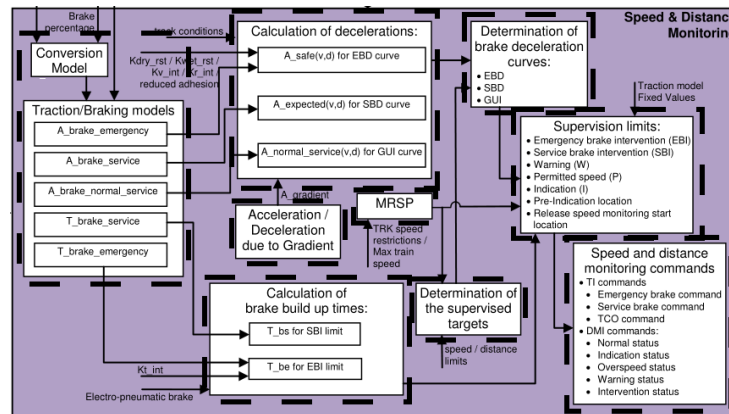
The borders for the subsystem decomposition are shown in Fig. 2. The dashed lines show the separate sub-machines. The dataflows that cross these lines are represented by the shared variables of the decomposed model.

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.

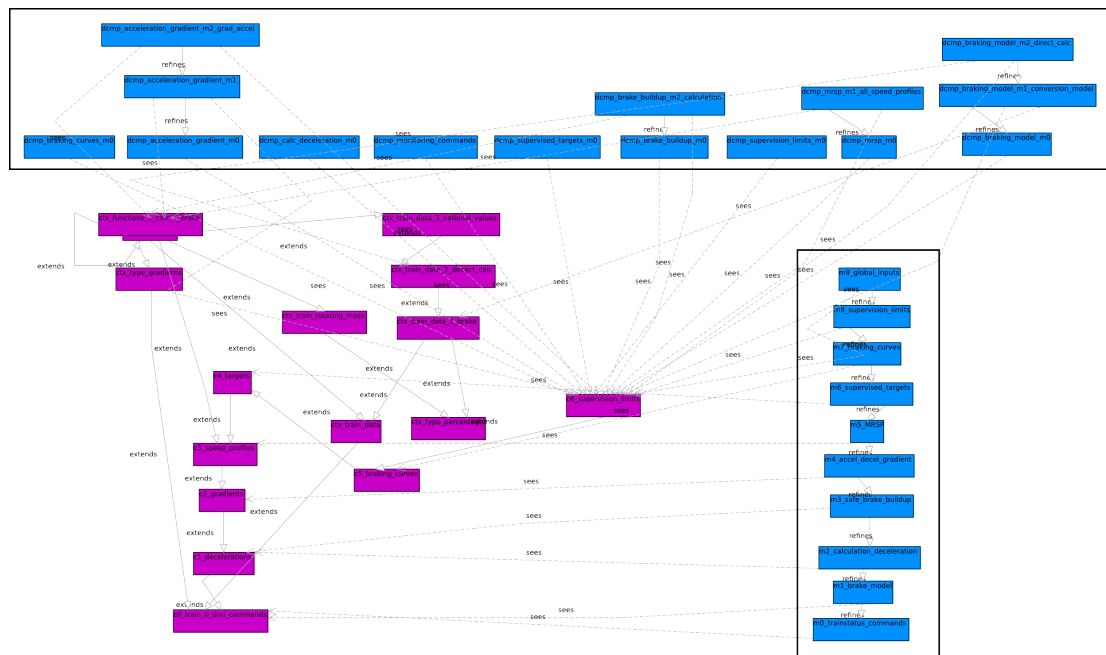
This refinement and context overview is very different from the others, as first an abstract global model was developed and then this model was decomposed into sub-models which are further refined. The contexts are shared between the decomposed models as far as possible. In this case, all resulting contexts 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.

<sup>1</sup>[http://wiki.event-b.org/index.php/Decomposition\\_Plug-in\\_User\\_Guide](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* ∈ *t\_speed*  
**axm4** : *c\_a0* ∈ *t\_acceleration*  
  
**axm5** : *c\_l0* ∈ *t\_locations*  
**axm6** : *c\_a\_brake0* ∈ *t\_speed* → *t\_acceleration*  
 default brake profile  
**axm7** : *c\_T\_brake0* ∈ *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* ∈ *t\_speed*  
**inv2** : *a\_current* ∈ *t\_acceleration*  
**inv3** : *loc\_current* ∈ *t\_locations*  
**inv4** : *cmd\_current* ∈ *t\_TI\_commands*  
**inv5** : *status\_current* ∈ *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 ≡
  any
    l_speed
    l_accel
    l_loc
  where
    grd1 : l_speed ∈ t_speed
    grd2 : l_accel ∈ t_acceleration
    grd3 : l_loc ∈ t_locations
  then
    act1 : v_current := l_speed
    act2 : a_current := l_accel
    act3 : loc_current := l_loc
  end
Event new_outputs ≡
  any
    l_ti_cmd
    l_dmi_status
  where
    grd1 : l_ti_cmd ∈ t_TI_commands
    grd2 : l_dmi_status ∈ 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 ≡
  any
    l_a_brake
  where
    grd1 : l_a_brake ∈ t_speed → t_acceleration
  then
    act1 : A_brake_emergency := l_a_brake
  end
Event set_A_brake_service ≡

```

```

    any
    where  $l\_a\_brake$ 
    then  $grd1 : l\_a\_brake \in t\_speed \rightarrow t\_acceleration$ 
    end  $act1 : A\_brake\_service := l\_a\_brake$ 
Event  $set\_A\_brake\_normal\_service \hat{=}$ 
    any
    where  $l\_a\_brake$ 
    then  $grd1 : l\_a\_brake \in t\_speed \rightarrow t\_acceleration$ 
    end  $act1 : A\_brake\_normal\_service := l\_a\_brake$ 
Event  $set\_T\_brake\_service \hat{=}$ 
    any
    where  $l\_T\_brake$ 
    then  $grd1 : l\_T\_brake \in t\_time$ 
    end  $act1 : T\_brake\_service := l\_T\_brake$ 
Event  $set\_T\_brake\_emergency \hat{=}$ 
    any
    where  $l\_T\_brake$ 
    then  $grd1 : l\_T\_brake \in t\_time$ 
    end  $act1 : T\_brake\_emergency := l\_T\_brake$ 
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
     $axm1 : f\_A\_deceleration0 \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
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  $\hat{=}$ 
    any

        l_a_decel
    where

        grd1 :  $l_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
        grd2 :  $A\_brake\_emergency \in t\_speed \rightarrow t\_acceleration$ 
    then

        act1 :  $A\_safe := l\_a\_decel$ 
    end
Event set_A_expected  $\hat{=}$ 
    any

        l_a_decel
    where

        grd1 :  $l_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
        grd2 :  $A\_brake\_service \in t\_speed \rightarrow t\_acceleration$ 
    then

        act1 :  $A\_expected := l\_a\_decel$ 
    end
Event set_A_normal_service  $\hat{=}$ 
    any

        l_a_decel
    where

        grd1 :  $l_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
        grd2 :  $A\_brake\_normal\_service \in t\_speed \rightarrow t\_acceleration$ 
    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 ≡
  any
    l_t_be
  where
    grd1 : l_t_be ∈ t_time
    grd2 : T_brake_emergency ∈ t_time
  then
    act1 : T_be := l_t_be
  end
Event set_T_bs ≡
  any
    l_t_bs
  where
    grd1 : l_t_bs ∈ t_time
    grd2 : T_brake_service ∈ 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 ≡
  any
    l_a_gradient
  where
    grd1 : l_a_gradient ∈ 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  $\hat{=}$ 
    any
    where
        l_sp
    then
        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  $\hat{=}$ 
    any
        where
             $l\_target$ 
            grd1 :  $l\_target \in t\_targets$ 
            grd2 :  $MRS P \in t\_speed\_profiles$ 
        then
            act1 :  $EOA := l\_target$ 
        end
Event set_LOA  $\hat{=}$ 
    any
        where
             $l\_target$ 
            grd1 :  $l\_target \in t\_targets$ 
            grd2 :  $MRS P \in t\_speed\_profiles$ 
        then
            act1 :  $LOA := l\_target$ 
        end
Event set_SvL  $\hat{=}$ 
    any
        where
             $l\_target$ 
            grd1 :  $l\_target \in t\_targets$ 
            grd2 :  $MRS P \in t\_speed\_profiles$ 
        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  $\hat{=}$ 
    any

        l_curve
        where

            grd1 :  $l\_curve \in t\_braking\_curves$ 
            grd2 :  $A\_safe \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd3 :  $A\_expected \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd4 :  $A\_normal\_service \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd5 :  $LOA \in t\_targets$ 
            grd6 :  $EOA \in t\_targets$ 
            grd7 :  $SvL \in t\_targets$ 

        then

            act1 :  $EBD := l\_curve$ 

        end
Event set_SBD  $\hat{=}$ 
    any

        l_curve
        where

            grd1 :  $l\_curve \in t\_braking\_curves$ 
            grd2 :  $A\_safe \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd3 :  $A\_expected \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd4 :  $A\_normal\_service \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd5 :  $LOA \in t\_targets$ 
            grd6 :  $EOA \in t\_targets$ 
            grd7 :  $SvL \in t\_targets$ 

        then

            act1 :  $SBD := l\_curve$ 

        end
Event set_GUI  $\hat{=}$ 
    any

        l_curve
        where

            grd1 :  $l\_curve \in t\_braking\_curves$ 
            grd2 :  $A\_safe \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd3 :  $A\_expected \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            grd4 :  $A\_normal\_service \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 
            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
     $PII$  pre-indication_location
     $RSM\_start$  release speed monitoring start location
EVENTS
Event  $set\_EBI \hat{=}$ 
    any

         $I\_limit$ 
    where

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

         $act1 : EBI := I\_limit$ 
    end
Event  $set\_SBI \hat{=}$ 
    any

         $I\_limit$ 
    where

```

```

    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 
    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  $\hat{=}$ 
  any
     $l\_limit$ 
  where
    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 
    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  $\hat{=}$ 
  any
     $l\_limit$ 
  where
    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 
    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  $\hat{=}$ 
  any
     $l\_limit$ 
  where
    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 
    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_Pll_limit  $\hat{=}$ 
  any
     $l\_limit$ 
  where
    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 

```

```

    grd3 :  $EBD \in t\_braking\_curves$ 
    grd4 :  $SBD \in t\_braking\_curves$ 
    grd5 :  $GVI \in t\_braking\_curves$ 
    grd6 :  $T\_bs \in t\_time$ 
    grd7 :  $T\_be \in t\_time$ 
  then
    act1 :  $PII := l\_limit$ 
  end
Event set_RSM_start_limit  $\hat{=}$ 
  any
    l_limit
  where
    grd1 :  $l\_limit \in t\_supervision\_limits$ 
    grd2 :  $MRS P \in t\_speed\_profiles$ 
    grd3 :  $EBD \in t\_braking\_curves$ 
    grd4 :  $SBD \in t\_braking\_curves$ 
    grd5 :  $GVI \in t\_braking\_curves$ 
    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.

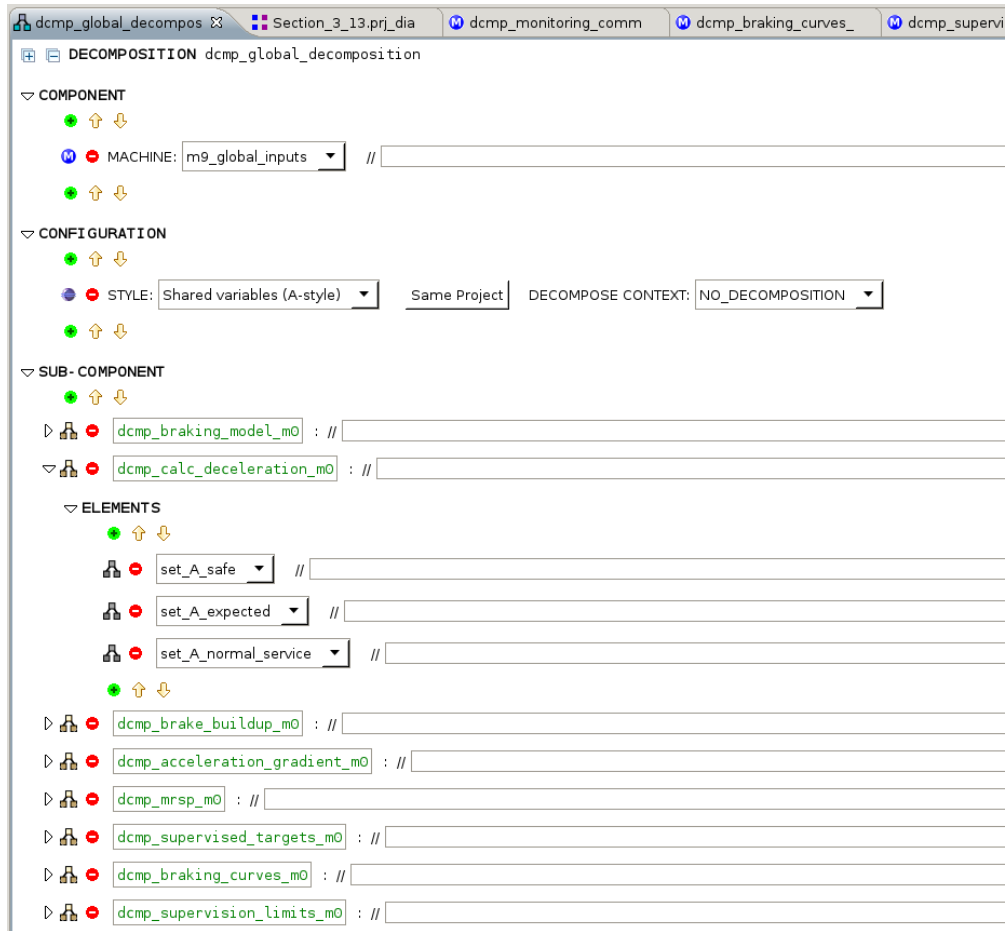


Figure 4. Decomposition Configuration

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  
 $T_{brake\_service}$  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**  $update\_train\_state \triangleq$  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 ≡
    any
        l_a_brake
    where
        grd1 : l_a_brake ∈ t_speed → t_acceleration
    then
        act1 : A_brake_emergency := l_a_brake
    end
Event set_A_brake_service ≡
    any
        l_a_brake
    where
        grd1 : l_a_brake ∈ t_speed → t_acceleration
    then
        act1 : A_brake_service := l_a_brake
    end
Event set_A_brake_normal_service ≡
    any
        l_a_brake
    where
        grd1 : l_a_brake ∈ t_speed → t_acceleration
    then
        act1 : A_brake_normal_service := l_a_brake
    end
Event set_T_brake_service ≡
    any
        l_T_brake
    where
        grd1 : l_T_brake ∈ t_time
    then
        act1 : T_brake_service := l_T_brake
    end
Event set_T_brake_emergency ≡
    any
        l_T_brake
    where
        grd1 : l_T_brake ∈ 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** :  $f\_conversion\_model\_applicable \in t\_speed \times t\_brake\_percentage \times t\_length \times t\_brake\_position \rightarrow BOOL$

cf. 3.13.3.2.1

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

cf. 3.13.3.3

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

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

cf. 3.13.3.4

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

**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*  $\hat{=}$



```

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  $\hat{=}$ 
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  $\hat{=}$ 
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
        c_train_length ↦ l_target_speed)
    end
Event specify_brake_percentage ≡
    any
        l_brake
    where
        grd1 : l_brake ∈ 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 ≡
    when
        then
            grd1 : brake_percentage_via_train_data = TRUE
        then
            act1 : brake_percentage := c_brake_percentage0
            act2 : brake_percentage_via_train_data := FALSE
        end
    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\_direct\_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 :  $f\_calc\_A\_brake\_service\_direct \in t\_brake\_combination \rightarrow (t\_speed \rightarrow t\_acceleration)$

axm3 :  $f\_calc\_A\_brake\_emergency\_direct \in t\_brake\_combination \rightarrow (t\_speed \rightarrow t\_acceleration)$

axm4 :  $f\_calc\_A\_brake\_normal\_direct \in t\_brake\_position \times t\_acceleration \times t\_acceleration$   
 $\rightarrow (t\_speed \rightarrow t\_acceleration)$

cf. 3.13.2.2.3.1.10

the accelerations are the  $c\_SB01$  and  $c\_SB02$  constants

```

    axm5 : c_SB01 ∈ t_acceleration
    axm6 : c_SB02 ∈ t_acceleration
    axm7 : f_calc_T_brake_service ∈ t_brake_combination → t_time
    axm8 : f_calc_T_brake_emergency ∈ t_brake_combination → 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_dirrect_calc
EVENTS
  Event calc_A_brake_emergency_direct ≡
  refines set_A_brake_emergency
    any
      where l_brake_combination
      with grd1 : l_brake_combination ∈ t_brake_combination
      then l_a_brake : l_a_brake = f_calc_A_brake_emergency_direct(l_brake_combination)
      end act1 : A_brake_emergency := f_calc_A_brake_emergency_direct(l_brake_combination)
    end
  Event calc_A_brake_service_direct ≡
  refines set_A_brake_service
    any
      where l_brake_combination
      with grd2 : l_brake_combination ∈ t_brake_combination
      then l_a_brake : l_a_brake = f_calc_A_brake_service_direct(l_brake_combination)
      end act1 : A_brake_service := f_calc_A_brake_service_direct(l_brake_combination)
    end
  Event set_A_brake_normal_service ≡
  refines set_A_brake_normal_service
    any
      where l_brake_position
      with grd1 : l_brake_position ∈ t_brake_position
      then l_a_brake : l_a_brake = f_calc_A_brake_normal_direct(l_brake_position ↦ c_SB01 ↦ c_SB02)
      end act1 : A_brake_normal_service := f_calc_A_brake_normal_direct(l_brake_position ↦ c_SB01 ↦ c_SB02)
    end
  Event set_T_brake_service ≡
  refines set_T_brake_service
    any

```

```

    where  $l\_brake\_combination$ 

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

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

    end  $act1 : T\_brake\_service := f\_calc\_T\_brake\_service(l\_brake\_combination)$ 
  end
Event  $set\_T\_brake\_emergency \hat{=}$ 
refines  $set\_T\_brake\_emergency$ 
  any
    where  $l\_brake\_combination$ 

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

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

    end  $act1 : T\_brake\_emergency := f\_calc\_T\_brake\_emergency(l\_brake\_combination)$ 
  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  $set\_A\_safe \hat{=}$ 
  any
    where  $l\_a\_decel$ 

    then  $grd1 : l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 

    end  $act1 : A\_safe := l\_a\_decel$ 
  end
Event  $set\_A\_expected \hat{=}$ 
  any
    where  $l\_a\_decel$ 

    then  $grd1 : l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 

    end  $act1 : A\_expected := l\_a\_decel$ 
  end
Event  $set\_A\_normal\_service \hat{=}$ 
  any
    where  $l\_a\_decel$ 

    then  $grd1 : l\_a\_decel \in t\_speed \times t\_distance \rightarrow t\_acceleration$ 

    end  $act1 : A\_normal\_service := l\_a\_decel$ 
  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 REFINES  
 $T\_bs$  Shared variable, DO NOT REFINES  
 $a\_current$  Shared variable, DO NOT REFINES  
 $v\_current$  Shared variable, DO NOT REFINES  
 $T\_brake\_service$  Shared variable, DO NOT REFINES  
 $loc\_current$  Shared variable, DO NOT REFINES  
 $T\_brake\_emergency$  Shared variable, DO NOT REFINES

**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**  $update\_train\_state \hat{=}$

External event, DO NOT REFINES

**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**  $set\_T\_brake\_service \hat{=}$

External event, DO NOT REFINES

**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  $\hat{=}$ 
    External event, DO NOT REFINE
    any
        where l_T_brake
        then grd1 : l_T_brake  $\in$  t_time
        then act1 : T_brake_emergency := l_T_brake
        end
Event set_T_be  $\hat{=}$ 
    any
        where l_t_be
        then grd1 : l_t_be  $\in$  t_time
        then act1 : T_be := l_t_be
        end
Event set_T_bs  $\hat{=}$ 
    any
        where l_t_bs
        then 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** :  $T\_brake\_service\_via\_train\_data \in \text{BOOL}$

**inv3** :  $T\_brake\_emergency\_data \in t\_time$

**inv4** :  $T\_brake\_emergency\_via\_train\_data \in \text{BOOL}$

**EVENTS**

**Event**  $set\_T\_be \hat{=}$

**refines**  $set\_T\_be$

**any**

$l\_brake\_combination$

**where**

**grd1** :  $T\_brake\_emergency\_via\_train\_data = \text{TRUE}$

**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**  $calc\_T\_be\_conversion\_model \hat{=}$

**refines**  $set\_T\_brake\_emergency$

**when**

**grd1** :  $T\_brake\_emergency\_via\_train\_data = \text{FALSE}$

**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**  $set\_T\_bs \hat{=}$

**refines**  $set\_T\_bs$

**any**

$l\_brake\_combination$

**where**

**grd1** :  $T\_brake\_service\_via\_train\_data = \text{TRUE}$

**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 ↦ l_brake_combination)
    end
Event calc_T_bs_conversion_model ≡
refines set_T_bs
    any
        where
            l_brake_combination
        with
            grd1 : T_brake_service_via_train_data = FALSE
            grd2 : l_brake_combination ∈ t_brake_combination
        then
            l_t_bs : l_t_bs = f_T_brake_emergency(T_brake_service ↦ l_brake_combination)
        end
        act1 : T_bs := f_T_brake_emergency(T_brake_service ↦ l_brake_combination)
    end
Event specify_T_brake_service_train_data ≡
    any
        where
            l_T_brake
        then
            grd1 : l_T_brake ∈ t_time
            grd2 : T_brake_service_via_train_data = FALSE
        end
        act1 : T_brake_service_data := l_T_brake
        act2 : T_brake_service_via_train_data := TRUE
    end
Event remove_T_brake_service_data ≡
    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 ≡
    any
        where
            l_T_brake
        then
            grd1 : l_T_brake ∈ t_time
            grd2 : T_brake_emergency_via_train_data = FALSE
        end
        act1 : T_brake_emergency_data := l_T_brake
        act2 : T_brake_emergency_via_train_data := TRUE
    end
Event remove_T_brake_emergency_data ≡
    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*  $\hat{=}$

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*  $\hat{=}$

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

(loc, SvL, compensated\_grad, mass, speed\_profile)  
 calculates acceleration due to gradient according to 3.13.4

**axm2** :  $f\_grad\_uphill \in t\_gradients \rightarrow BOOL$

indicates whether gradient is uphill or downhill

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

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

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

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 ∈ BOOL
    inv2 : M_rotating_nom ∈ t_mass_percentage
EVENTS
Event set_A_gradient ≡
refines set_A_gradient
    any
        l_mass
        l_SvL
        l_grad
        l_TSR
    where
        grd2 : l_mass ∈ t_mass_percentage
        grd3 : l_SvL ∈ t_locations
        grd4 : l_grad ∈ t_gradients
        grd5 : l_TSR ∈ t_speed_profiles
    with
        l_a_gradient : l_a_gradient = f_accel_due_gradient(loc_current ↦ l_SvL ↦ l_grad ↦ l_mass ↦ l_TSR)
    then
        act1 : A_gradient := f_accel_due_gradient(loc_current ↦ l_SvL ↦ l_grad ↦ l_mass ↦ l_TSR)
    end
Event specify_rotating_mass ≡
    any
        l_mass
    where
        grd1 : rotating_mass_specified = FALSE
        grd2 : l_mass ∈ t_mass_percentage
    then
        act1 : M_rotating_nom := l_mass
        act2 : rotating_mass_specified := TRUE
    end
Event delete_rotating_mass_data ≡
    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*  $\hat{=}$ **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*  $\hat{=}$ **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*  $\hat{=}$ **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**

*MRS P* Shared variable, DO NOT REFIN

*a\_current* Shared variable, DO NOT REFIN

*v\_current* Shared variable, DO NOT REFIN

*loc\_current* Shared variable, DO NOT REFIN

**INVARIANTS**

*typing\_MRS P* : *MRS P*  $\in$  *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* : *MRS P* := *c\_speed\_profile0*

**end**

**Event** *update\_train\_state*  $\hat{=}$

External event, DO NOT REFIN

**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*  $\hat{=}$

**any**

*l\_sp*

**where**

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

**then**

*act1* : *MRS P* := *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*  $\hat{=}$

**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* : *MRS P* := *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  $\hat{=}$ 
    any
        l_target
    where
        grd1 : l_target  $\in$  t_targets
    then
        act1 : EOA := l_target
    end
Event set_LOA  $\hat{=}$ 
    any
        l_target
    where
        grd1 : l_target  $\in$  t_targets
    then
        act1 : LOA := l_target
    end
Event set_SvL  $\hat{=}$ 
    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 emergency 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  $\hat{=}$ 
    any
        l_curve
    where
        grd1 : l_curve  $\in$  t_braking_curves
    then
        act1 : EBD := l_curve
    end

```

```

    end
  Event set_SBD ≡
    any
      where
        l_curve
      then
        grd1 : l_curve ∈ t_braking_curves
      end
      act1 : SBD := l_curve
    end
  Event set_GUI ≡
    any
      where
        l_curve
      then
        grd1 : l_curve ∈ t_braking_curves
      end
      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* ∈ *t\_supervision\_limits*  
 typing\_I\_limit : *I\_limit* ∈ *t\_supervision\_limits*  
 typing\_Pil : *Pil* ∈ *t\_supervision\_limits*  
 typing\_W\_limit : *W\_limit* ∈ *t\_supervision\_limits*  
 typing\_RSM\_start : *RSM\_start* ∈ *t\_supervision\_limits*  
 typing\_P\_limit : *P\_limit* ∈ *t\_supervision\_limits*  
 typing\_EBI : *EBI* ∈ *t\_supervision\_limits*

**EVENTS**

```

  Event set_EBI ≡
    any
      where
        l_limit
      then
        grd1 : l_limit ∈ t_supervision_limits
      end
      act1 : EBI := l_limit
    end

```



```

    end
Event set_SBI  $\hat{=}$ 
  any
    l_limit
  where
    grd1 : l_limit  $\in$  t_supervision_limits
  then
    act1 : SBI := l_limit
  end
Event set_W_limit  $\hat{=}$ 
  any
    l_limit
  where
    grd1 : l_limit  $\in$  t_supervision_limits
  then
    act1 : W_limit := l_limit
  end
Event set_I_limit  $\hat{=}$ 
  any
    l_limit
  where
    grd1 : l_limit  $\in$  t_supervision_limits
  then
    act1 : I_limit := l_limit
  end
Event set_P_limit  $\hat{=}$ 
  any
    l_limit
  where
    grd1 : l_limit  $\in$  t_supervision_limits
  then
    act1 : P_limit := l_limit
  end
Event set_PII_limit  $\hat{=}$ 
  any
    l_limit
  where
    grd1 : l_limit  $\in$  t_supervision_limits
  then
    act1 : PII := l_limit
  end
Event set_RSM_start_limit  $\hat{=}$ 
  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*  $\hat{=}$

**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 Gdemann. Event-B Model of Subset 026, Section 3.5. github - openETCS, 2013.
- [Mat13b] Matthias Gdemann. Event-B Model of Subset 026, Section 4.6. github - openETCS, 2013.
- [Mat13c] Matthias Gdemann. 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.