Using ERTMSFormalSpecs to model ERTMS braking curves

Laurent Ferier (laurent@ertmssolutions.com)
ERTMS SOLUTIONS – Brussels

Svitlana Lukicheva (svitlana@ertmssolutions.com)
ERTMS SOLUTIONS – Brussels

Stanislas Pinte (stan@ertmssolutions.com)
ERTMS SOLUTIONS – Brussels

The European Railway Traffic Management System (ERTMS) defines standards for interoperability between the on-board train protection systems (ETCS) and the railway infrastructure. Part of this specification describes the computation of train braking curves and the associated train reactions according to its speed. This document explains how ERTMSFormalSpecs has been used to model such braking curves and the associated advantages of traceability and understandability of using a domain-specific language as opposed to a more common specification-implementation process, where multiple stages of human interpretation and interaction increase the opportunities for errors.

1 The Context

The European Railway Traffic Management System (ERTMS) defines the standard for interoperability between the on-board train protection systems (ETCS) and the railway infrastructure. This standard comprises several documents (referred to as Subsets) focusing on specific parts of the system.

The ERTMSFormalSpecs (EFS) is a project that focuses on the modelling of the requirements expressed in Subset-026 – System Requirements Specification related to the trainborne equipment, see /2./, allowing for automated code generation, and applying the tests specified in Subset-076 – Test cases on the model, see /3./. More information about EFS can be found in /1./.

This document describes the modelling effort associated with the specification of the braking curves as specified in Chapter 3 of Subset-026. This document covers:

- The location principles, train position and train orientation
- The concept of MA (Movement Authority)
- Determination of the most restrictive speed profile (MRSP) according to static speed restrictions, such as static speed profile, axle load speed profile, temporary speed restrictions, speed restrictions related to signaling, mode, train, level crossing
- Computation of the deceleration and brake build up time, according to the brake configuration of the train
- Acceleration or deceleration due to gradients
• According to the MA, MRSP, deceleration factors related to the train brakes and the gradient, determination of the targets and brake deceleration curves

• Supervision limits such as
  o ceiling supervision limits which allow to define the Emergency Brake Intervention (EBI) curve, Service Brake Intervention (SBI) and Warning (W) based on the Permitted speed
  o target supervision limits

• According to these limits, Subset-026 describes the speed and distance monitoring algorithms which allow the on-board to generate braking commands, traction cut-off commands and display relevant information to the driver.

![Figure 1 - EFS inputs and outputs](image)

Figure 1 describes the computation of the deceleration curves is based on the train characteristics and the trackside conditions that the train must satisfy. These data are dependent on the odometry:

• Nominal distance: as soon as a point of interest has been reached by the train, the corresponding speed restriction is removed from the computation of the maximum permitted speed

• Estimated speed: some speed restrictions are dependent on the current speed

• Current time: to evaluate the timers associated to an MA.

The output of the computation is deceleration curves (EBD, SBD and GUI), expressed as a sequence of parabolic curves which never exceed the maximum permitted speed. These curves shall be used by the on-board equipment in order to generate braking commands described in the Chapter 3.13.10 of the Subset-026 v320.

The train and trackside related inputs are described in the Chapter 3 of the Subset-026, whereas description of the different packets is provided by the Chapter 7.
2 Braking curves computation

The braking curves computation requires to compute several artifacts as described in the following sections:

- Determination of the permitted speed and associated speed limitation curves.
- Deceleration factors according to gradients and train brakes.
- Determination of the deceleration according to the permitted speed and deceleration factors.
- Determination of the target supervision limits and according to these limits, determination of the on-board commands.

2.1 Permitted speed and limitation curves

The permitted speed (P) as defined in Subset-026, Section 3.13.9.2 is the base function which provides the train permitted speed, in terms of a distance:

\[ P(Distance) : Speed \]

It is a discontinuous step function which can be described as a sequence of constant speed values. An example is shown in Figure 2, where the Permitted speed step function is displayed in blue, provides a speed (on the Y axis) based on the train location, distance on the X axis.

![Figure 2 - Permitted speed](image)

As we shall see in this section, the computation of the permitted speed is performed based on the MA and the current MRSP, which in turn is computed in terms of static speed restrictions, such as SSP, TSRs, LXs, …

2.1.1 Single elements of the Permitted speed

For each element which composes P, Chapter 7 of Subset-026 describes the messages used to configure such elements (for instance Packet 65: Temporary Speed Restriction) and the interpretation to be given to such packets (for instance, Temporary Speed Restrictions are described in Section 3.11.5).

Each of the static speed profile speed restrictions has been modeled with a step function

\[ SpeedRestriction(Distance) : Speed \]
For instance, the speed restriction function related to a TSR has been modeled in EFS language as follow:

```plaintext
// Provides the speed profile due to a temporary speed restriction.
// -------------------------
FUNCTION TSRSpeedRestriction(
aTSR: TemporarySpeedRestriction
Distance: BaseTypes.Distance
) RETURNS BaseTypes.Speed

// During TSR
aTSR.Location <= Distance AND Distance < aTSR.Location + TSRLength (aTSR) =>
aTSR.Speed

// Outside TSR
=> BaseTypes.Speed.MaxSpeed
```

The function TSRSpeedRestriction() indicates that, when the distance passed as parameter lies between the boundaries of the TSR, the maximum speed must correspond to the speed defined in that TSR, whereas everywhere outside those boundaries, the corresponding speed is not limited.

It interprets data from packet 65, as described in Subset-026, Section 7.4.2.17 (see Figure 3),

<table>
<thead>
<tr>
<th>Description</th>
<th>Transmission of temporary speed restriction.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted by</td>
<td>Any</td>
</tr>
<tr>
<td>Content</td>
<td>Variable</td>
</tr>
<tr>
<td>NID_PACKET</td>
<td>8</td>
</tr>
<tr>
<td>Q_DIR</td>
<td>2</td>
</tr>
<tr>
<td>L_PACKET</td>
<td>13</td>
</tr>
<tr>
<td>Q_SCALE</td>
<td>2</td>
</tr>
<tr>
<td>NID_TSR</td>
<td>8</td>
</tr>
<tr>
<td>D_TSR</td>
<td>15</td>
</tr>
<tr>
<td>L_TSR</td>
<td>15</td>
</tr>
<tr>
<td>Q_FRONT</td>
<td>1</td>
</tr>
<tr>
<td>V_TSR</td>
<td>7</td>
</tr>
</tbody>
</table>

**Figure 3 - Packet 65: Temporary Speed Restriction**

by decoding D_TSR according to packet Q_SCALE to the attribute Location. The length of the TSR is provided by TSRLength() according to the train length and Q_FRONT.

```plaintext
// Provides the length of the TSR together with the train length, if needed.
// -------------------------
FUNCTION TSRLength(
aTSR : TemporarySpeedRestriction
) RETURNS BaseTypes.Length

// TrainLengthDelay
aTSR.TrainLengthDelay ==
Messages.Q_FRONT.Train_length_delay_on_validity_end_point_of_profile_element =>
aTSR.Length + Kernel.TrainData.TrainData.Value.TrainLength

// NoTrainLengthDelay
=> aTSR.Length
```
2.1.2 Combining all compounds elements
Since more than one TSR can be configured at a time in the system, the EFS interpreter combines all the temporary speed restrictions by minimizing them (according to Subset-026 3.11.5.4),

> 3.11.5.4 When two or more temporary speed restrictions overlap, the most restrictive speed of the overlapping temporary speed restrictions shall be used in the area of overlap.

This is formally defined by

```plaintext
//Provides the speed restrictions profile due to TSR.
//----------------------------------------
FUNCTION SpeedRestrictions(
    Distance: BaseTypes.Distance)
RETURNS BaseTypes.Speed

// Value
=> (REDUCE TSRs USING MIN (First => FUNCTION d : BaseTypes.Distance =>
        TSRSpeedRestriction(aTSR => X, Distance => d),
    Second => RESULT) INITIAL_VALUE BaseTypes.MaxSpeedFunction)(Distance)
```

where the REDUCE operator iterates over TSRs, the collection of active TSR, and applies the MIN function between the current TSR and what has already been computed. Initially (when no TSR has been considered), there is no speed restriction, hence, the result is equal to the maximum speed.
This function is used by the EFS interpreter to create the speed restriction related to all TSRs step functions as displayed in Figure 5.

Figure 5 - Combined TSRs
### 2.1.3 Computing the MRSP

The same process is applied to compute the MRSP step function, as defined in 3.13.7.2:

3.13.7.2 The Most Restrictive Speed Profile shall be computed from all speed restrictions (see 3.13.2.13 & 3.13.2.3.2) by selecting the most restrictive parts of each element, some elements being compensated by the train length if requested by trackside (see 3.11.3.1.3 for SSP, 3.11.4.6 for ASP and 3.11.5.3 for TSR).

Since the train length is already taken into account for SSP, ASP and TSR, the MRSP is simply computed by minimizing the static speed profile, the axle load speed profile, temporary speed restrictions, speed restrictions related to STMs, mode, train, level crossing. This is formally defined by

```plaintext
// Provides the speed restrictions associated to the MRSP
FUNCTION SpeedRestrictions(distance: Default.BaseTypes.Distance)
  RETURNS Default.BaseTypes.Speed
// Always
  => MIN(
    MIN(
      LX.SpeedRestrictions,
      MIN(
        TSR.SpeedRestrictions,
        NationalValues.ModeRelatedSpeedRestrictions
      )
    ),
    MIN(
      MIN(
        NationalValues.OverrideRelatedSpeedRestrictions,
        TrainData.TrainData.Value.MaximumSpeed
      ),
      MIN(
        STM.MaxSpeedRestrictions,
        STM.SystemSpeedRestrictions
      )
    )
  ),
  MIN(
    AxleLoad.SpeedRestrictions,
    StaticSpeedProfile.SpeedRestrictions
  )(distance)
```

---

Using ERTMSFormalSpecs to model ERTMS braking curves
The EFS interpreter creates the corresponding step function, for instance see Figure 6 for an example of an MRSP.

2.1.4 Computing the Permitted speed

The permitted speed is defined in Subset-026 by Section 3.11.1.1:

3.11.1 The permitted speed at which the train is allowed to travel shall be limited to different kinds of Static Speed Restrictions and considering the End of Authority/Limit Of Authority.

The different kinds of static restrictions are defined by the MRSP whereas the End Of Authority / Limit Of Authority relates to the MA. Hence, the permitted speed is computed by minimizing those two speed restrictions:

```plaintext
// The speed restriction applied to the train. This combines both MA and MRSP speed restrictions
//-------
FUNCTION P(
  distance: BaseTypes.Distance
) RETURNS BaseTypes.Speed
// Always
=> MIN(MA.SpeedRestrictions, MRSP.SpeedRestrictions)(distance)
```
2.1.5 Traceability and understandability

This approach allows to easily check that the model corresponds to the specification, since the model is written in a functional language, very close to the mathematical language and the model elements are traced to the corresponding requirements from Subset-026.

Moreover, the step functions can be used to understand the reasons why the speed, at a given location, is limited to given value. For instance, this traceability information is shown, in Figure 7. The blue function displays the MRSP, whereas we have selected also to display in orange, the speed restrictions related to level crossing and in green, the speed restrictions related to the TSR. Thanks to this graph, we easily understand why the MRSP is limited to 40 km/h at 2000m (due to a Level crossing, displayed in orange) and limited to 25 km/h between 4000m and 6500m (due to a TSR, displayed in green).

![Figure 7 - MRSP, Level crossing and TSRs](image)

2.2 Deceleration factors

The deceleration functions used in the deceleration curve computation (such as A_safe, A_expected and A_normal_service) as defined in Subset-026, Section 3.13.6 are functions which provide the train deceleration based on the train speed and distance:

\[
\begin{align*}
A_{\text{safe}}(\text{Distance}, \text{Speed}) & : \text{Acceleration} \\
A_{\text{expected}}(\text{Distance}, \text{Speed}) & : \text{Acceleration} \\
A_{\text{normal\_service}}(\text{Distance}, \text{Speed}) & : \text{Acceleration}
\end{align*}
\]

These functions are defined based on the acceleration/deceleration due to the gradient and the deceleration due to the several brakes configuration. The EFS interpreter can determine the deceleration step surfaces, that is a step function which takes two parameters, the same way the most restrictive speed step functions were determined, as described in Section 2.1.
For instance, Figure 8 presents the speed deceleration surface of A\_brake\_emergency computed for a specific train configuration and based on a gradient profile received from trackside. Each box in the graph represents a specific deceleration based on the train position (represented on the X axis) and a train speed.

![Figure 8 - A\_brake\_emergency(Distance, Speed) deceleration surface](image)

### 2.3 Deceleration curves

The deceleration curves (EBD, SBD and GUI) can be computed based on the permitted speed and the deceleration factors according to Section 3.13.8 of Subset-026.

#### 2.3.1 Ceiling speed limits

The deceleration curves EBD, SBD and GUI are computed based on the permitted speed $P$, on which ceiling speed limits have been applied (3.13.9.2), by respectively adding $dV_{ebi}$, $dV_{sbi}$ and $dV_{warning}$, a function which gives a delta Speed based on a distance. For instance, $dV_{ebi}$ is described in Subset-026 section 3.13.9.2.3.

3.13.9.2.3 For $dV_{ebi}$, the following formula shall be applied:

$$
\text{when } V_{MBSP} > V_{ebi\_min}:
\begin{align*}
    dV_{ebi} &= \min\{dV_{ebi\_min} + C_{ebi} \cdot (V_{MBSP} - V_{ebi\_min}), dV_{ebi\_max}\} \\
    \text{with } C_{ebi} &= \frac{(dV_{ebi\_max} - dV_{ebi\_min})}{(V_{ebi\_max} - V_{ebi\_min})}
\end{align*}
$$

\text{when } V_{MBSP} \leq V_{ebi\_min}:
\begin{align*}
    dV_{ebi} &= dV_{ebi\_min}
\end{align*}
Using ERTMSFormalSpecs to model ERTMS braking curves

which is in turn modeled by

```plaintext
/// The delta speed to add to P in order to get EBI
///
/// FUNCTION dv_ebi(  
/// V: BaseTypes.Speed  
/// ) RETURNS BaseTypes.Speed
///
/// // V > V_ebi_min  
/// V > V_ebi_min =>  
/// BaseTypes.MinSpeed(dV_ebi_min + C_ebi() * (V - V_ebi_min), dV_ebi_max)
///
/// // V <= V_ebi_min  
/// => dV_ebi_min
///
/// // C_ebi
/// //
/// FUNCTION C_ebi() RETURNS BaseTypes.Speed
/// // Always
/// => (dV_ebi_max - dV_ebi_min) / (V_ebi_max - V_ebi_min)
```

One can easily ensure that this model corresponds to its requirement.

Based on this function, the function EBI, as described in 3.13.9.2.2

3.13.9.2.2 From an MRSP element or from the LOA, the Permitted speed, Warning, Service brake intervention and Emergency brake intervention supervision limits are defined (see Figure 43)

is modeled by

```plaintext
/// Provides EBI on the permitted speed restriction
///
/// FUNCTION EBI(  
/// distance : BaseTypes.Distance  
/// ) RETURNS BaseTypes.Speed
///
/// // Permitted speed is not zero  
/// P ( distance ) > 0.0 =>  
/// AddIncrement(Function => P, Increment => CeilingSupervision.dv_ebi)(distance)
///
/// // Otherwise
/// => 0
```

http://www.ertmssolutions.com
This gives the following EBI function displayed in blue, based on the permitted speed displayed in red on Figure 9. This figure allows to easily show that the EBI effectively corresponds to its requirements.

2.3.2 Deceleration curves

The EFS interpreter computes, based on a maximum speed function and a deceleration factor, the deceleration curve as requested in Section 3.13.8.1.3 of Subset-026. This function is modelled by

```plaintext
// Provides the deceleration profile based on EBI and the A_safe deceleration factor
----------------
FUNCTION EBD(
    Distance : BaseTypes.Distance
) RETURNS BaseTypes.Speed

// Always => DecelerationProfile(
    SpeedRestrictions => EBI,
    DecelerationFactor => A_Safe)(Distance)
```

The EFS Interpreter determines the supervised targets and computes the interconnected arcs of parabola according to these targets and the deceleration factors encountered as noted in Section 3.13.6.2.1.9 of Subset-026.
2.4 Target supervision limits

The target supervision limits are computed by considering the first distance for which a specific speed is encountered on a deceleration curve. For instance, $d_{SBI1}$, as defined in Section 3.13.9.3.3.1 of Subset-026 is modeled by

```plaintext
//Provides the distance $d_{SBI1}$
//----------------
FUNCTION d_SBI1(
  Vest : Default.BaseTypes.Speed
) RETURNS Default.BaseTypes.Distance

// Always
=> DistanceForSpeed(Function => SBD, Speed => Vest) + Vest * Tbs1
```

Where DistanceForSpeed(SBD, Vest) provides the first distance where the curve SBD reaches the speed Vest.

3 Conclusions

We have shown in this paper how the EFS environment has been used to compute braking curves as specified in document Subset-026. We have shown that the language is perfectly suited to describe the requirements and close enough to them to allow a domain expert to assess that the model is correctly implemented. Moreover, the way we modelled the several speed restrictions allows a domain expert to understand, for a specific situation, the reason why a specific speed restriction applies.

We also have shown that the language is formal enough to allow the computation of deceleration curves based on all the defined model elements and compute target supervision limits, interpreted when used within the workbench and allowing for automated code generation to produce production-level implementations, where these limits are used to generate braking commands.

4 Further works

All target supervision limits have not been modelled yet. As soon as all those limits are modelled, we shall model the speed and distance monitoring commands to generate braking commands, traction cut-off commands and relevant information to the driver using the state machines available in the EFS environment.
5 Bibliography
The following documents are referenced inside that document:

/1./ ERTMS Formal Specs: a domain specific language to formalize ERTMS specifications for onboard unit development
/2./ [Subset-026] Version 3.2.0
/3./ [Subset-076] Version 1.0.2

6 Version History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/9/2011</td>
<td>1</td>
<td>First version</td>
</tr>
<tr>
<td>14/09/2011</td>
<td>2</td>
<td>Integrated comments from Owen O'Connor (<a href="mailto:owen.oconnor@ntlworld.com">owen.oconnor@ntlworld.com</a>)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fixed typos and grammar</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reviewed the figures to keep the same scale on each one of them.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Reviewed Figure 8 to display the deceleration value for each one of the tiles.</td>
</tr>
<tr>
<td>2/10/2011</td>
<td>3</td>
<td>Integrated comments from Rob van Looveren (<a href="mailto:rovanlooveren@hotmail.com">rovanlooveren@hotmail.com</a>)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Figure 1 explaining EFS inputs and outputs</td>
</tr>
</tbody>
</table>