OETCS/WP4/CSM – 01/00

COMPASS

ITEA2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

ITEA2 Project
Call 6 11025
2012 – 2015

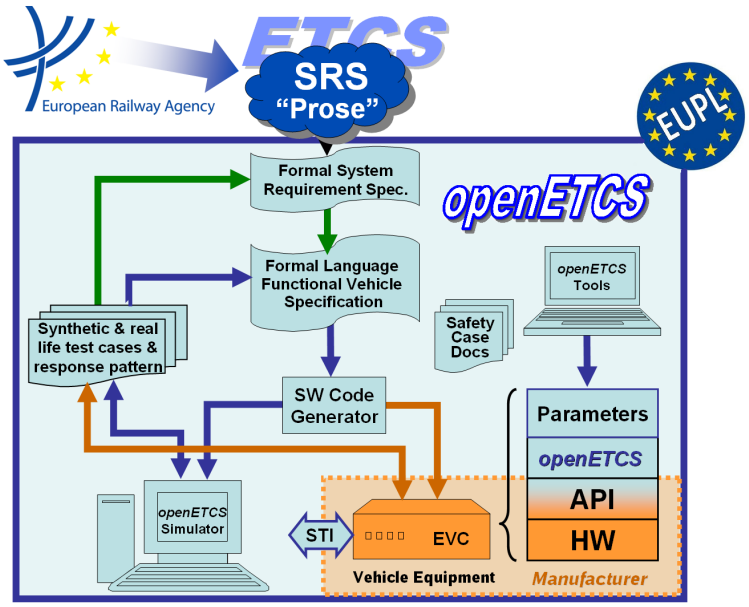
Work Package 4

A SysML Test Model and Test Suite for the ETCS Ceiling Speed Monitor

Technical report

Cécile Braunstein, Jan Peleska *, Uwe Schulze*, Felix Hübner **,
Wen-ling Huang*, Anne E. Haxthausen*** and Linh Vu Hong***

2014-05-11



Funded by:

Federal Ministry of Education and Research

Région de Bruxelles-Capitale

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

COMPASS

RobustRailS

openETCS

This page is intentionally left blank

Work Package 4 **OETCS/WP4/CSM – 01/00**
2014-05-11

A SysML Test Model and Test Suite for the ETCS Ceiling Speed Monitor

Technical report

Cécile Braunstein, Jan Peleska *, Uwe Schulze* and Felix Hübner **

University Bremen

Wen-ling Huang*

University of Hamburg and University of Bremen, Germany

Anne E. Haxthausen*** and Linh Vu Hong***

DTU Computer Technical University of Denmark

- * The authors' research is funded by the EU FP7 COMPASS project under grant agreement no.287829.
- ** The author's research is funded by Siemens in the context of the SyDE Graduate School on System Design
- *** The authors' research is funded by the RobustRailS project funded by the Danish Council for Strategic Research.

Technical Report

Prepared for openETCS@ITEA2 Project

Abstract: In this technical report a detailed model description of a train control system application is given. The application consists of the ceiling speed monitoring (CSM) function for the European Vital Computer which is the main onboard controller for trains conforming to the European Train Control System specification. The model is provided in SysML, and it is equipped with a formal semantics that is consistent with the (semi formal) SysML standard published by the Object Management Group (OMG). The model and its description are publicly available on http://www.mbt-benchmarks.de, a website dedicated to the publication of models that are of interest for the model-based testing (MBT) community, and may serve as benchmarks for comparing MBT tool capabilities. The model described here is of particular interest for analysing the capabilities of equivalence class testing strategies. The CSM application inputs velocity values from a domain which could not be completely enumerated for test purposes with reasonable effort. We describe a novel method for equivalence class testing that – despite the conceptually infinite cardinality of the input domains – is capable to produce finite test suites that are exhaustive under certain hypotheses about the internal structure of the system under test.

Keywords Model-based testing, Equivalence class partition testing, UML/SysML, European Train Control System ETCS, Ceiling Speed Monitoring

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

1.1 A Test Model for the ETCS Ceiling Speed Monitor

In 2011 the *model-based testing benchmarks website* www.mbt-benchmarks.org has been created with the objective to publish test models that may serve as challenges or benchmarks for validating testing theories and for comparing the capabilities of model-based testing (MBT) tools [24]. In this technical report a novel contribution to this website is presented, a SysML model of the Ceiling Speed Monitor (CSM) which is part of the European Vital Computer (EVC), the onboard controller of trains conforming to the European Train Control System (ETCS) standard [6]. In Section 2 the functional objectives for the CSM are described, and in Section 3 the detailed model description is provided.

The static and behavioural semantics of SysML models have been defined in [21, 22] in a semi-formal way, leaving certain "semantic variation points" open, so that they can be adjusted according to project-specific requirements. For automated model-based testing, however, a strictly formal semantics is required, so that concrete test data can be calculated from the model's transition relation using constraint solving techniques [12]. We therefore describe in Section 4 how a formal behavioural semantics is derived for the CSM model and present the associated transition relation in propositional form.

We use state transition systems (STS) for encoding the operational semantics of concrete modelling formalisms like SysML. STS are widely known from the field of model checking [5], because their extension into Kripke Structures allows for effective data abstraction techniques. The latter are also applied for equivalence class testing. Since state transition systems are a means for semantic representation, testing theories elaborated for STS are applicable for all concrete formalisms whose behavioural semantics can be expressed by STS. In [13] it is shown how the semantics of general SysML models and models of a process algebra are encoded as STS. In this technical report we illustrate how this is achieved for the concrete case of the CSM SysML model.

1.2 Equivalence Class Partition Testing for the CSM

The CSM represents a specific test-related challenge: its behaviour depends on actual and allowed speed, and these have conceptually real-valued data domains, so that – even when discretising the input space – it would be infeasible to exercise all possible combinations of inputs on the system under test (SUT). Therefore equivalence class partition (ECP) testing strategies have to be applied for testing the CSM. While these strategies are well-adopted in a heuristic manner in today's industrial test campaigns, practical application of equivalence class testing still lacks formal justification of the equivalence classes selected and the sequences of class representatives selected as test cases: standard text books used in industry, for example [27], only explain the generation of input equivalence class tests for systems, where the SUT reaction to an input class representative is independent on the internal state. Moreover, the systematic calculation of classes from models, as well as their formal justification with respect to test strength and coverage achieved, is not yet part of today's best practices in industry.

In contrast to this, formal approaches to equivalence class testing have been studied in the formal methods communities; references to these results are given in Section 11. In the second main part of this report (Section 5) we therefore describe a recent formal technique for equivalence class testing and its application to testing the CSM. The theoretical foundations of this strategy have been published by two of the authors in [12]. This technical report illustrates its practical

application and presents first evaluation details using a prototype implementation in an existing MBT tool; the ECP tests are compared to test results obtained when applying other MBT coverage strategies, such as transition cover or MC/DC coverage (Section 10).

1.3 Fault Models and Completeness Results

Our ECP strategy introduces test suites depending on *fault models*. This well adopted notion has first been introduced in the field of finite state machine (FSM) testing [25], but is also applicable to other formal modelling techniques. A fault model consists of a reference model, a conformance relation and a fault domain. The latter is a collection of models whose behaviour may or may not be consistent to the reference model in the sense of the conformance relation. The test suites generated by the ECP strategy described here are *complete* with respect to the given fault model: each system of the fault domain which conforms to the reference model will pass all the generated tests (this means that the test suite is *sound*), and each system in the fault domain that violates the conformity to the reference model will fail at least once when tested according to the test suite (the test suite is *exhaustive*).

2 The Ceiling Speed Monitoring Function – Functional Objectives

The European Train Control System ETCS relies on the existence of an onboard controller in train engines, the *European Vital Computer EVC*. Its functionality and basic architectural features are described in the public ETCS system specification [6]. One functional category of the EVC covers aspects of speed and distance monitoring, to accomplish the *"... supervision of the speed of the train versus its position, in order to assure that the train remains within the given speed and distance limits."* [28, 3.13.1.1]. Speed and distance monitoring is decomposed into three sub-functions [28, 3.13.10.1.2], where only one out of these three is active at a point in time:

1. *Ceiling speed monitoring (CSM)* supervises the observance of the maximal speed allowed according to the current most restrictive speed profile (MRSP). CSM is active while the train does not approach a target (train station, level crossing, or any other point that must be reached with predefined speed).
2. *Target speed monitoring (TSM)* supervises the observance of the maximal distance-depending speed, while the train brakes to a target, that is, a location where a given predefined speed (zero or greater zero) must be met.
3. *Release speed monitoring (RSM)* applies when the special target "end of movement authority (EOA)" is approached, where the train must come to a stop. RSM supervises the observance of the distance-depending so-called release speed, when the train approaches the EOA.

In this technical report we present a complete formal model of the CSM function, with the objective to derive a complete test suite from this model (Section 5).

3 Model Description

3.1 Model Availability

The ceiling speed monitor has been modelled using the *OMG Systems Modeling Language (OMG SysML^{TM})* [21]. The complete model is available for download under `http://www.mbt-benchmarks.org`. This is a website dedicated to the publication of test models possessing features that are of general interest for researchers and practitioners in the field of model-based testing (MBT). Moreover,

the models may serve as benchmarks for comparing test automation tools with respect to test strength and tool performance. This has been further motivated in [24], where also suggestions for MBT benchmarks are given. In this section, we give a comprehensive introduction into the formal model.

3.2 Model Components

According to the model-based testing approach applied in this report, UML/SysML test models are structured into the following basic components.

1. A package containing the system requirements (Fig. 3),
2. A block diagram (Fig. 1) identifying
   - the system under test (SUT),
   - its interface to the operational environment, and
   - the test environment (TE) simulating the "real" operational environment during test execution,
3. Subordinate block diagrams refining the internal structure of the SUT (Fig. 2) and the TE, respectively,
4. State machines associated with the leaf blocks of the structural decompositions of SUT (Fig. 4) and TE, and
5. Operations associated with blocks. These are referenced by state machines, when evaluating guard conditions or performing actions.

3.3 Model Semantics – Overview

The detailed formal behavioural semantics of SysML test models has been described in [13, pp. 88]. This semantics is consistent with the standards [22, 21], but fixes certain semantic variation points in ways that are admissible according to the standards. In this technical report, however, the semantic details will be explained as far as they are relevant for understanding the model presented here: in the present section, the behavioural semantics of the CSM model is informally explained, and in Section 4, the formal model semantics is specified by presenting its transition relation.

The leaf components of the structural model decomposition execute concurrently. For the model under consideration the SUT operates in a sequential manner, but concurrently with its environment. In this test model, the behaviour of the TE is undetermined; this is interpreted in the way that every possible sequence of input vectors to the SUT would be allowed. This assumption is reasonable for the example considered here: due to robustness requirements, the ceiling speed monitor must be able to cope with input sequences that may be unreasonable from a physical point of view. TE components are introduced in situations where only certain types of interactions between operational environment and SUT are possible.

The model executes according to the *run-to-completion* semantics defined for state machines in [22]. The model is in a *quiescent* (or stable) state, if no transition can be executed without an input change. In a quiescent model state, inputs may be changed. If these changes enable a transition, the latter is executed. Since our SUT model is deterministic – this is typical for

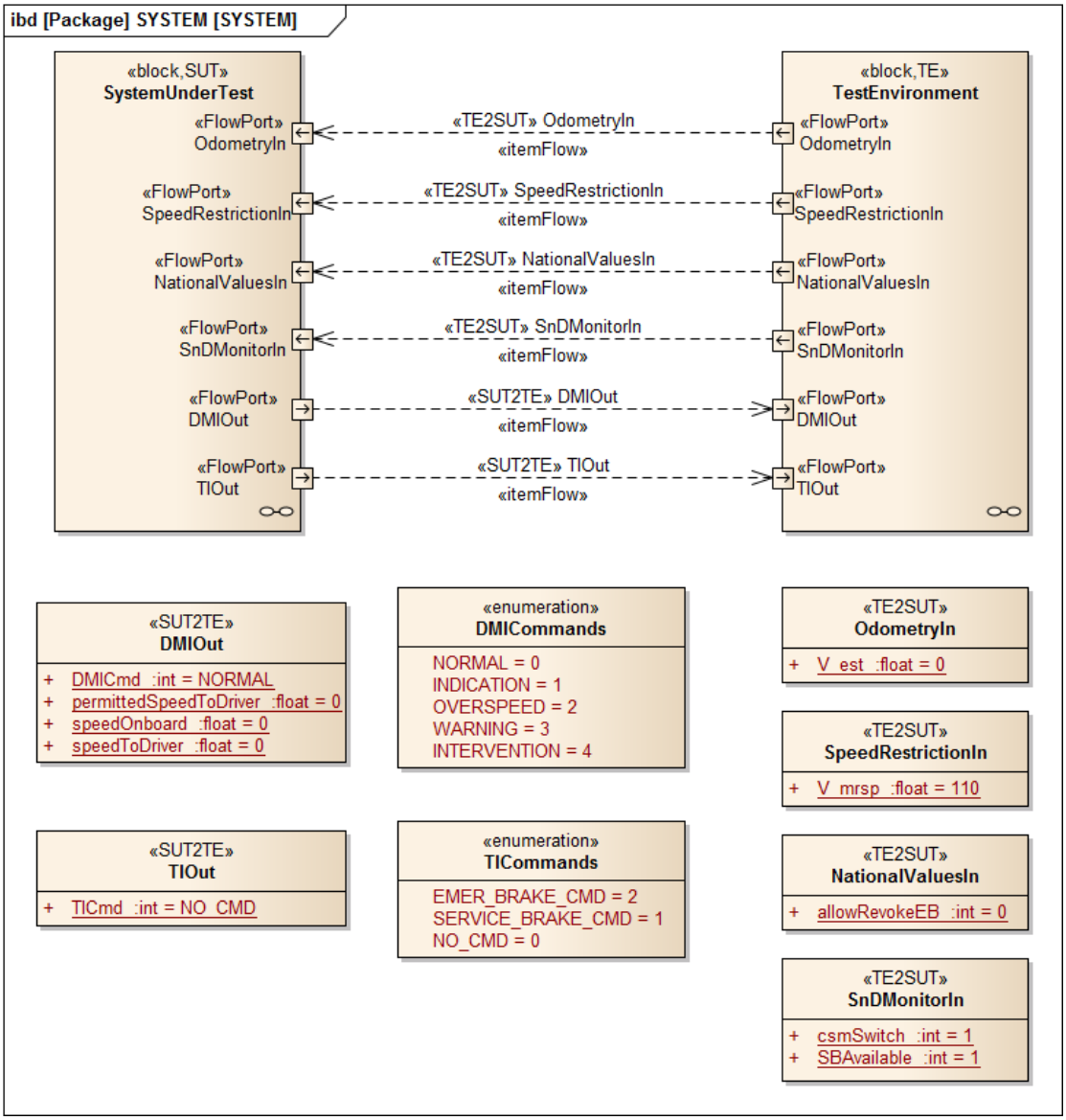


Figure 1. System interface of the ceiling speed monitor.

safety-relevant applications – there is no necessity to handle situations where several transitions are simultaneously enabled. The executed transition, however, may lead to a *transient* state, that is, to a state where another transition is enabled. In the run-to-completion semantics this new transition is also executed, and so forth until a quiescent state is reached. Conceptually, the consecutive execution of model transitions is executed in zero time, so that input changes cannot happen until the next quiescent state has been reached. Moreover, models admitting unbounded sequences of transitions between transient states are considered as illegal, and this situation is called a *livelock* failure.

3.4 Interfaces

The interfaces between SUT and its environment are specified in the internal block diagram displayed in Fig. 1. All interfaces are represented as flow ports. The environment writes to SUT input ports and reads from SUT output ports.

Ceiling speed monitoring is activated and de-activated by the speed and distance monitoring (SnD) coordination function that controls CSM, TSM, and RSM: on input interface SnDMonitorIn,

variable csmSwitch specifies whether ceiling speed monitoring should be active (csmSwitch = 1) or passive, since target or release speed monitoring is being performed (csmSwitch = 0). Furthermore, this interface carries variable SBAvailable which has value 1, if the train is equipped with a service brake. This brake is then used for slowing down the train if it has exceeded the maximal speed allowed, but not yet reached the threshold for an emergency brake intervention. If SBAvailable = 0, the emergency brake shall be used for slowing down the train in this situation. Input SBAvailable is to be considered as a configuration parameter of the train, since it depends on the availability of the service brake hardware. Therefore this value can be freely selected at start-of-test, but must remain constant during test execution.

Input interface OdometryIn provides the current speed value estimated by the odometer equipment in variable V_{est}. Input interface SpeedRestrictionIn provides the current maximal velocity defined by the most restrictive speed profile in variable V_{MRSP}. Input interface NationalValuesIn provides a control flag for the ceiling speed monitor: variable allowRevokeEB is 1, if after an emergency brake intervention the brake may be automatically released as soon as the estimated velocity of the train is again less or equal to the maximal speed allowed. Otherwise (allowRevokeEB = 0) the emergency brakes must only be released after the train has come to a standstill ($V_{est} = 0$). This input parameter is called a "national value", because it may change when a train crosses the boundaries between European countries, due to their local regulations.

Output interface DMIOut sends data from the SUT to the driver machine interface (DMI). It carries five variables. DMICmd is used to display the supervision status to the train engine driver: Value INDICATION may be initially present when CSM is activated, but will be immediately overridden by one of the values NORMAL, OVERSPEED, WARNING, or INTERVENTION, as soon as ceiling speed monitoring becomes active. Value NORMAL is written by the SUT to this variable as long as the ceiling speed is not violated by the current estimated speed. Value OVERSPEED has to be set by the CSM as soon as condition $V_{MRSP} < V_{est}$ becomes true. If the speed increases further (the detailed conditions are described below), the indication changes from OVERSPEED to WARNING, and from there to INTERVENTION. The latter value indicates that either the train is slowed down until it is back in the normal speed range, or the emergency brake has been triggered to stop the train. Furthermore, interface DMIOut contains the following speed-related variables that are displayed as y/t-diagrams on the DMI.

- speedToDriver: the current estimated speed as given by variable V_{est}.
- permittedSpeedToDriver: the permitted maximal speed as given by the most restrictive speed profile V_{MRSP}.
- speedOnBoard: maximal speed allowed (V_{MRSP}) as long as the train does not overspeed. Otherwise it carries values $V_{MRSP}+\delta$, where $\delta > 0$ specifies the margin from V_{MRSP} to service brake intervention and is calculated as described below.

Output interface TIout specifies the train interface from the CSM to the brakes, using variable TICmd. If TICmd = NO_CMD, both service brakes (if existent) and emergency brakes are released. If TICmd = SERVICE_BRAKE_CMD, the service brake is activated. If TICmd = EMER_BRAKE_CMD, the emergency brake is triggered.

3.5 SUT Attributes and Operations

The CSM is modelled as an application with sequential behaviour. Therefore the SUT block on the top-level interface diagram (Fig. 1) is refined into another block diagram that just carries the SUT, as shown in Fig. 2.

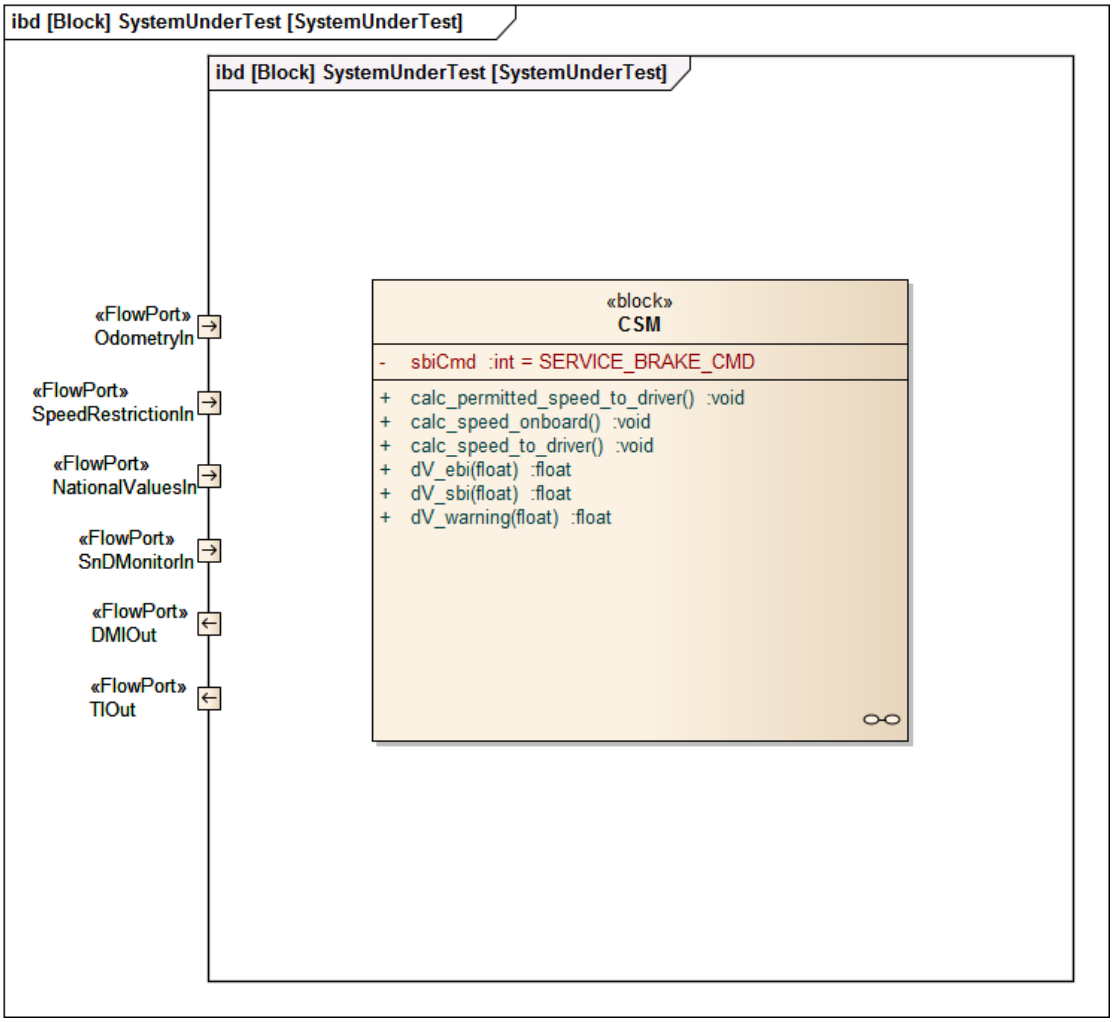


Figure 2. Block diagram with CSM (sequential behaviour).

As shown there, the SUT uses a local attribute sbiCmd which carries value SERVICE_BRAKE_CMD, if the service brake should be used for slowing down the train to the admissible speed. If the value EMER_BRAKE_CMD is assigned to sbiCmd, the emergency brake will be triggered in this situation.

Operations dV_warning(float), dV_sbi(float), dV_ebi(float) return values that are used to determine whether a warning should be indicated to the train engine driver ($V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$), a service brake intervention should be triggered ($V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$), or the emergency brake should be activated ($V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$). In each case, the calculation is performed according to the pattern

$$\mathrm{dV}_x(V_{MRSP}) = \begin{cases} \min\{\mathrm{dV}_{x\min} + C_x \cdot (V_{MRSP} - \mathrm{V}_{x\min}), \mathrm{dV}_{x\max}\} \\ \qquad \text{if } V_{MRSP} > \mathrm{V}_{x\min} \\ \mathrm{dV}_{x\min} \\ \qquad \text{if } V_{MRSP} \leq \mathrm{V}_{x\min} \end{cases} \tag{1}$$

which has been defined in [28, 3.13.9.2.3]. Here x can be replaced by warning, sbi (SBI = service brake intervention), and ebi (EBI = Emergency Brake Intervention), and C_x is defined by

$$C_x = \frac{\mathrm{dV}_{x\max} - \mathrm{dV}_{x\min}}{\mathrm{V}_{x\max} - \mathrm{V}_{x\min}}$$

The following minimal and maximal values apply [28, A.3.1]:

$\mathrm{dV}_{\mathsf{warning}\min} = 4$	$\mathrm{dV}_{\mathsf{sbi}\min} = 5.5$	$\mathrm{dV}_{\mathsf{ebi}\min} = 7.5$
$\mathrm{dV}_{\mathsf{warning}\max} = 5$	$\mathrm{dV}_{\mathsf{sbi}\max} = 10$	$\mathrm{dV}_{\mathsf{ebi}\max} = 15$
$\mathrm{V}_{\mathsf{warning}\min} = 110$	$\mathrm{V}_{\mathsf{sbi}\min} = 110$	$\mathrm{V}_{\mathsf{ebi}\min} = 110$
$\mathrm{V}_{\mathsf{warning}\max} = 140$	$\mathrm{V}_{\mathsf{sbi}\max} = 210$	$\mathrm{V}_{\mathsf{ebi}\max} = 210$

Inserting these values into Equation (1) results in

$$\mathrm{dV}_{\mathsf{warning}}(V_{MRSP}) = \begin{cases} \min\{\frac{1}{3} + \frac{1}{30} \cdot V_{MRSP}, 5\} & \text{if } V_{MRSP} > 110 \\ 4 & \text{if } V_{MRSP} \leq 110 \end{cases} \tag{2}$$

$$\mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) = \begin{cases} \min\{0.55 + 0.045 \cdot V_{MRSP}, 10\} & \text{if } V_{MRSP} > 110 \\ 5.5 & \text{if } V_{MRSP} \leq 110 \end{cases} \tag{3}$$

$$\mathrm{dV}_{\mathsf{ebi}}(V_{MRSP}) = \begin{cases} \min\{-0.75 + 0.075 \cdot V_{MRSP}, 15\} & \text{if } V_{MRSP} > 110 \\ 7.5 & \text{if } V_{MRSP} \leq 110 \end{cases} \tag{4}$$

Operations calc_speed_to_driver() and calc_permitted_speed_to_driver() support the display of the current estimated speed and the maximum speed, respectively, at the driver machine interface by performing assignments to output variables:

$$\text{speedToDriver} = V_{est}$$
$$\text{permittedSpeedToDriver} = V_{MRSP}$$

Operation calc_speed_onboard() displays the maximal speed V_{MRSP} specified by the most restrictive speed profile in DMI interface variable speedOnBoard, as long as the train is not overspeeding. As soon as $V_{est} > V_{MRSP}$, this function calculates the service brake intervention speed and displays it via speedOnBoard, that is,

$$\text{speedOnBoard} = V_{MRSP} + \text{dV}_{\text{sbi}}(V_{MRSP})$$

where $\text{dV}_{\text{sbi}}(V_{MRSP})$ is calculated according to Equation 3.

3.6 Requirements

Figure 3 shows the requirements reflected by the model. The requirement labels refer to the sections of the ETCS standard document [28], from where they have been imported into the model. To make this technical report sufficiently self-contained, we list the requirements applicable to CSM in Table 1, and adapt the wording and the cross references to the technical report.

In requirement REQ-3.13.10.2.2, the traction cut-off command on the train interface is not explicitly addressed in our model, because it will always be triggered in synchrony with a braking command. We assume the existence of a driver software layer in the EVC that automatically triggers traction cut-off if

- a traction cut-off interface is implemented for the EVC, and
- a service brake or emergency brake command is issued.

Requirement REQ-3.13.10.2.3 states that national values can only influence the usage of the service brake when in TSM. We will therefore assume that the availability of the service brake and its use for slowing down the train when the emergency braking condition is not yet fulfilled is constant (i.e., SBAvailable = 0 or SBAvailable = 1) during CSM operation.

Requirement REQ-3.13.10.3.3 is described by two tables (see Table 2 and Table 3 below), it is then decomposed into sub-requirements REQ-3.13.10.3.3.t1, . . . , REQ-3.13.10.3.3.r1, each of them representing one line of these two tables.

Requirement REQ-3.13.10.3.4 is represented as a transition table, it is decomposed into sub-requirements, one for each relevant cells of the table (see Table 4).

Requirement REQ-3.13.10.3.7 is “delegated” to the surrounding software of the CSM: it is assumed in our model that the input V_{MRSP} is always set by the CSM software environment in a way that takes into account the min safe front end of the train.

3.7 Behavioural Specification

The behaviour of the ceiling speed monitor is modelled by the hierarchic state machine that is associated with the SUT block of Fig. 2 and displayed in Fig. 4 (top-level state machine) and Fig. 5 (lower-level state machine associated with composite state CSM_ON.

The top-level state machine controls activation and de-activation of the CSM. As soon as input variable csmSwitch on interface SnDMonitorIn gets value 1, the CSM is activated, and it is de-activated when csmSwitch falls back to 0. On activation, the auxiliary variable sbiCmd is set to EMER_BRAKE_CMD, if the input variable SBAvailable carries value 0, indicating

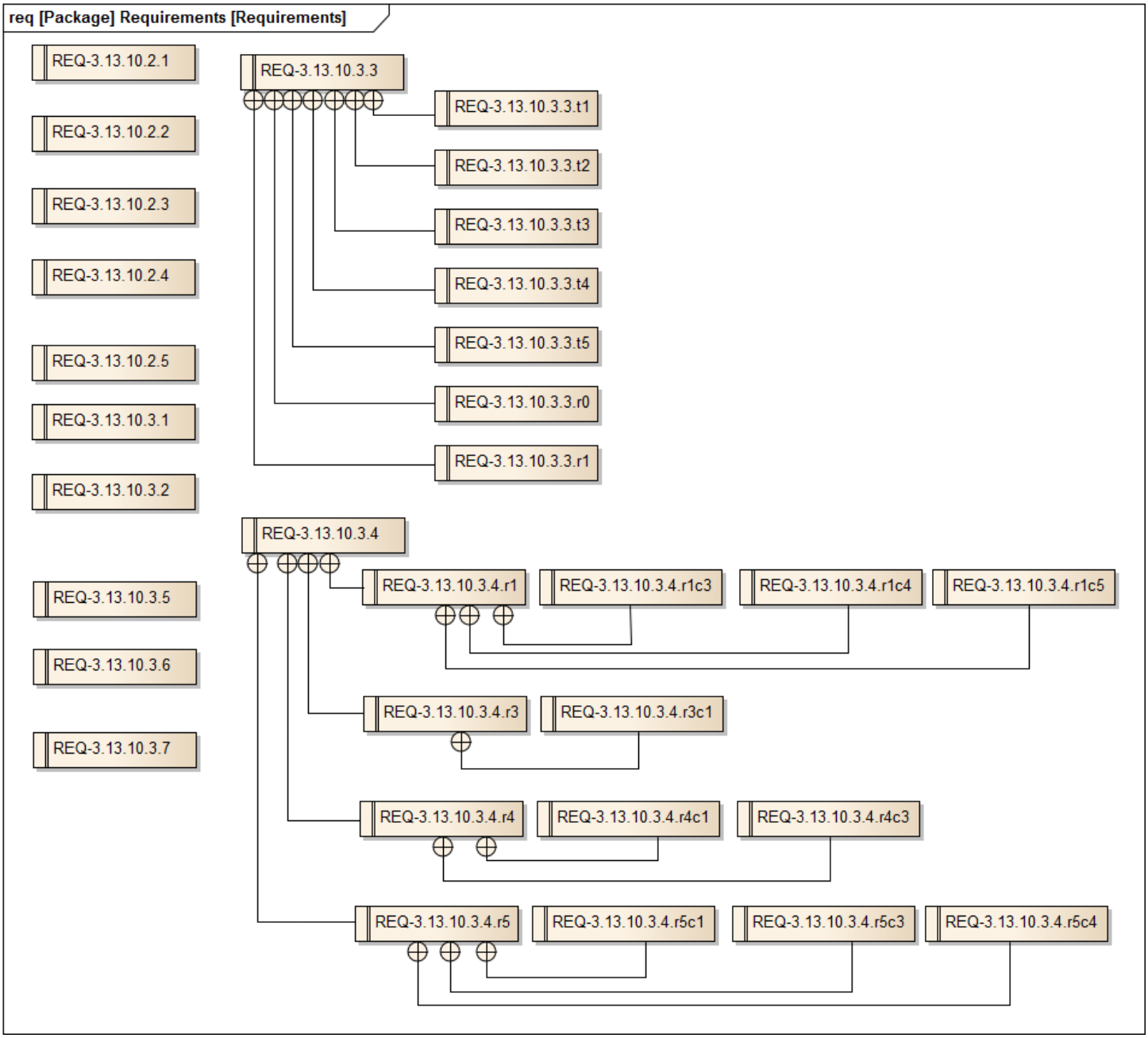


Figure 3. System requirements diagram.

Table 1. Requirements for the ceiling speed monitoring function.

id	Description
REQ-3.13.10.2.1	The train speed indicated to the driver shall be identical to the speed used for the speed monitoring (i.e. the estimated speed V_{est}).
REQ-3.13.10.2.2	Once a Train Interface command (traction cut-off, service brake or emergency brake) is triggered, the on-board shall apply it until its corresponding revocation condition is met.
REQ-3.13.10.2.3	If there is no on-board interface with the service brake or if the use of the service brake command is not allowed by a National Value (only in Target speed monitoring),whenever a service brake command is specified, the emergency brake command shall be triggered instead.
REQ-3.13.10.2.4	The emergency brake command, which is triggered instead of the service brake command when an SBI supervision limit is exceeded, shall be revoked according to the requirements specified for the revocation of service brake command, unless the emergency brake command has been also triggered due to an EBI supervision limit. In such case, the condition for revoking the emergency brake command due to EBI supervision limit shall prevail.
REQ-3.13.10.2.5	The on-board shall revoke the Intervention status only when no brake command is applied by the speed and distance monitoring function.
REQ-3.13.10.3.1	The on-board equipment shall display the permitted speed (V_{MRSP}).
REQ-3.13.10.3.2	When the supervision status is Overspeed, Warning or Intervention, the on-board equipment shall display the SBI speed (i.e. the FLOI speed; FLOI = First Line of Intervention).
REQ-3.13.10.3.3	The on-board shall compare the estimated speed with the ceiling supervision limits defined in [28, 3.13.9.2] and shall trigger/revoke commands to the train interface (service brake if implemented or emergency brake) and supervision statuses as described in Table 2 (from [28, Table 5]) and Table 3 (from [28, Table 6]).
REQ-3.13.10.3.4	The on-board equipment shall execute the transitions between the different supervision statuses as described in Table 4 (see [28, 4.6.1] for details about the symbols). This table takes into account the order of precedence between the supervision statuses and the possible updates of the MRSP while in ceiling speed monitoring (e.g. when a TSR is revoked; TSR = Temporary Speed Restriction).
REQ-3.13.10.3.5	When the speed and distance monitoring function becomes active and the ceiling speed monitoring is the first one entered, the triggering condition t1 defined in Table 2 shall be checked in order to determine whether the Normal status applies. If it is not the case, the on-board shall immediately set the supervision status to the relevant value, applying a transition from the Normal status according to Table 4.
REQ-3.13.10.3.6	The Indication status is not used in ceiling speed monitoring. However, in case the ceiling speed monitoring is entered and the supervision status was previously set to Indication, the on-board equipment shall immediately execute one of the transitions from the Indication status, as described in Table 4.
REQ-3.13.10.3.7	The locations corresponding to a speed increase of the MRSP shall be supervised by the on-board equipment taking into account the min safe front end of the train.

Table 2. Triggering of Train Interface commands and supervision statuses in ceiling speed monitoring (from [28, Table 5]).

id	TC	Estimated speed	TI	SSE
REQ-3.13.10.3.3.t1	t1	$V_{est} \leq V_{MRSP}$	—	Normal Status
REQ-3.13.10.3.3.t2	t2	$V_{est} > V_{MRSP}$	—	Overspeed Status
REQ-3.13.10.3.3.t3	t3	$V_{est} > V_{MRSP} + dV_{warning}$	—	Warning Status
REQ-3.13.10.3.3.t4	t4	$V_{est} > V_{MRSP} + dV_{sbi}$	SB	Intervention Status
REQ-3.13.10.3.3.t5	t5	$V_{est} > V_{MRSP} + dV_{ebi}$	EB	Intervention Status

TC: trigger condition
TI: command triggered on train interface to brakes
SB: trigger service brake command (if available, otherwise trigger emergency brake)
EB: trigger emergency brake command SSE: supervision status entered

Table 3. Revocation of Train Interface commands and supervision statuses in ceiling speed monitoring (from [28, Table 6]).

id	RC	Estimated Speed	TICR	SSR
REQ-3.13.10.3.3.r0	r0	Standstill	EB	Intervention Status
REQ-3.13.10.3.3.r1	r1	$V_{est} \leq V_{MRSP}$	SB, EB[a]	Indication Status Overspeed Status Warning Status Intervention Status (if SBI) Intervention Status (if EB and allowRevokeEB = 1)

[a]Only if allowRevokeEB = 1.

RC: revocation condition
TICR: command revoked on train interface to brakes
SSR: supervision status revoked

Table 4. Transitions between supervision statuses in ceiling speed monitoring (from [28, Table 7]).

Normal Status	$< r1$	$< r1$	$< r1$	$< r0, r1$
	Indication Status			
$t2 >$	$t2 >$	Overspeed Status		
$t3 >$	$t3 >$	$t3 >$	Warning Status	
$t4, t5 >$	$t4, t5 >$	$t4, t5 >$	$t4, t5 >$	Intervention Status

The sub-requirements IDs associated with each cell in the transition table are of the form REQ-3.13.10.3.4.rX.cY where X and Y are the row and column indexes, respectively.

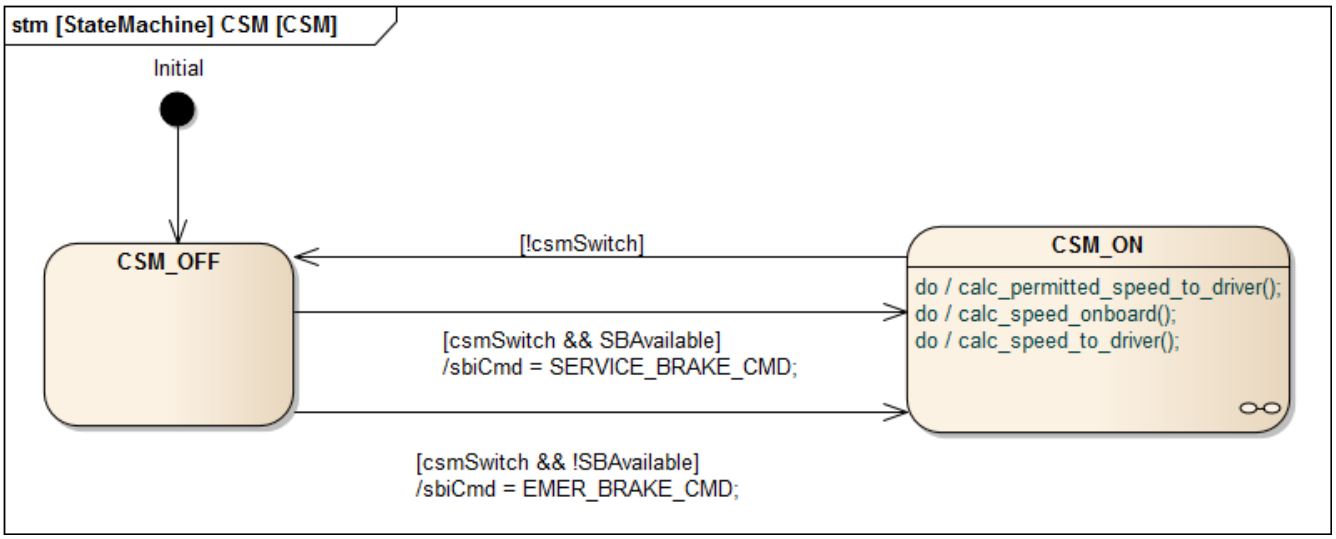


Figure 4. Ceiling speed monitoring – top-level state machine.

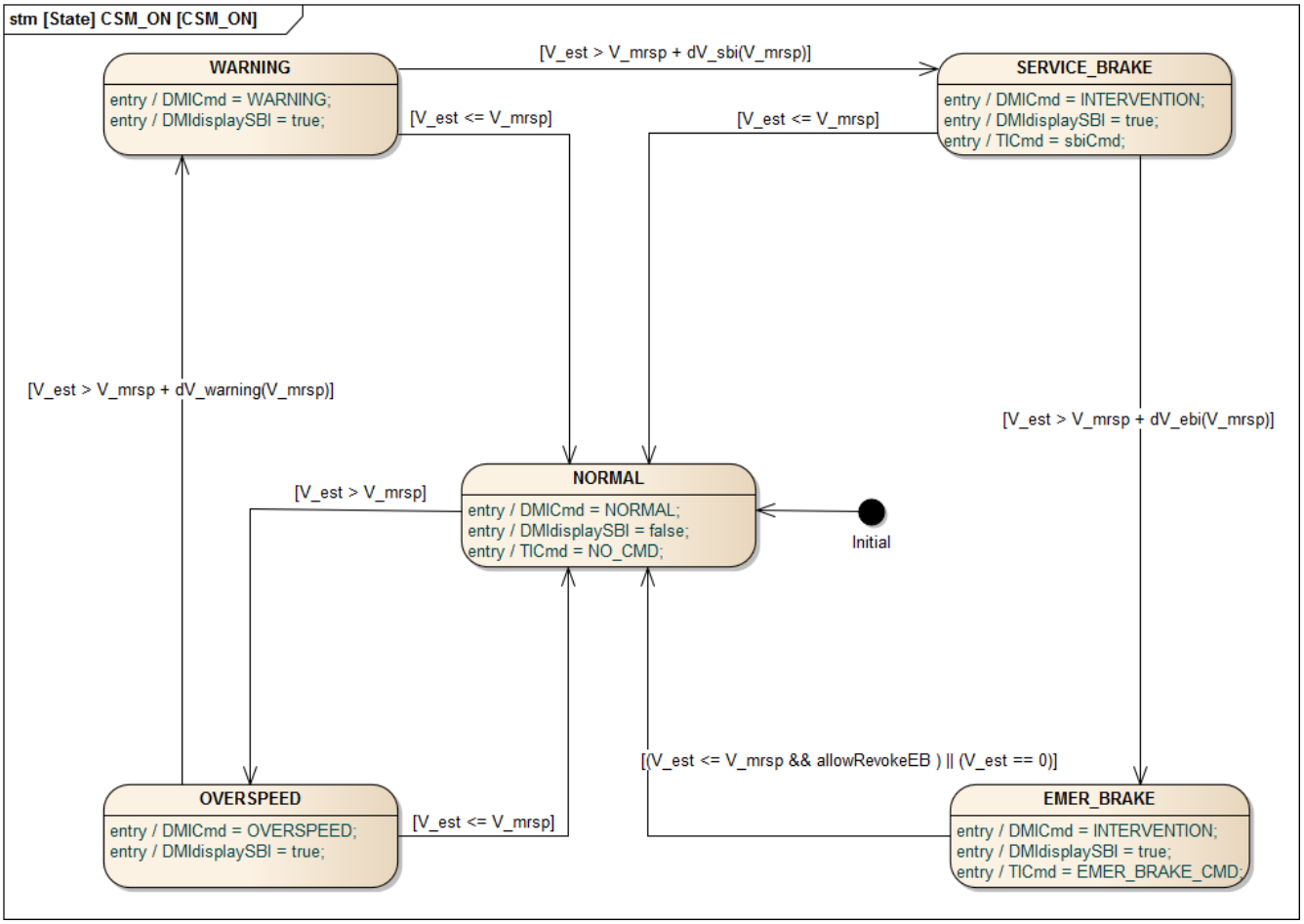


Figure 5. Ceiling speed monitoring state machine.

that no separate service brake can be used for slowing down the train, so that the emergency brake has to be used for this purpose. Conversely, when SBAvailable = 1, sbiCmd is set to SERVICE_BRAKE_CMD.

While in composite state CSM_ON, do actions are executed as specified by operations calc_permitted_speed_to_driver(), calc_speed_onboard(), and calc_speed_to_driver() introduced above. The effect of these do actions is that variables permittedSpeedToDriver, speedOnBoard, and speedToDriver are set consistently to the current values depending on V_{est} and V_{MRSP}, respectively, as described above. The do-actions are executed whenever all state machine transitions are blocked, and the value of the left-hand side variable of the assignment performed by one of these operations differs from the valuation of the right-hand side expression. As a consequence, the necessary updates of these three output variables are executed in zero time after any input change.

Subordinate state machine CSM_ON specifies the detailed behaviour of the CSM. Its execution starts in basic state NORMAL, where the 'NORMAL' indication is displayed on the DMI and brakes are released (TICmd = NO_CMD). When the speed increases above the maximal speed allowed ($V_{est} > V_{MRSP}$), the state machine transits to basic state OVERSPEED, where the 'OVERSPEED' indication is displayed to the train engine driver. If the train continues overspeeding until the warning threshold $V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$ is exceeded, a transition into the WARNING state is performed, accompanied by an indication change on the DMI. Accelerating further until $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$ leads to a transition into basic state SERVICE_BRAKE, where either the service brake or the emergency brake is triggered, depending on the value stored before in variable sbiCmd. The DMI display changes to 'INTERVENTION'.

The intervention status is realised by two basic state machine states, SERVICE_BRAKE and EMER_BRAKE. From SERVICE_BRAKE it is still possible to return to NORMAL, as soon as the speed has been decreased below the overspeeding threshold. When the train, however, continues its acceleration until the emergency braking threshold has been exceeded ($V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$), basic state EMER_BRAKE is entered. From there, a state machine transition to NORMAL is only possible if the train comes to a standstill, or if the national regulations (variable allowRevokeEB) allow to release the brakes as soon as overspeeding has stopped.

Observe that the run-to-completion semantics of state machines also allows for zero-time transitions from, for example, NORMAL to EMER_BRAKE. If, while in basic state NORMAL, the inputs change such that $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$ becomes true[1], the state machine transition from NORMAL to OVERSPEED leads to a transient model state, because guard condition $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$ is already fulfilled, and the state machine transits to WARNING. Similarly, guards $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$ and $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$ also evaluate to true, so that the next quiescent state is reached in basic state EMER_BRAKE. Therefore REQ-3.13.10.3.4.r5c1 which requires direct transitions from Normal status to Intervention status is fulfilled by the CSM_ON state machine: if the guard conditions have the appropriate valuations, the required target states can be reached in zero time, that is, in one observable EVC processing cycle. Analogously, the state machine fulfils requirements REQ-3.13.10.3.4.r5c2, REQ-3.13.10.3.4.r5c3, REQ-3.13.10.3.4.r4c1 without needing direct state machine transition arrows between the respective state machine states.

[1] This would be an exceptional behaviour situation, caused, for example, by temporary unavailability of odometry data, so that a "sudden jump" of V_{est} would be observed by the CSM.

3.8 Requirements Tracing

SysML provides language elements for relating model elements to requirements, using the «satisfy» relationship from model elements to requirements symbols in arbitrary SysML diagrams [21, Section 16]. Exploiting this language feature supports

- model validation, and
- requirements-based testing.

In the former case, missing requirements can be detected if they cannot be linked to structural or behavioural model elements in the appropriate way. In latter case, execution traces through the model covering a given structural or behavioural model element represent test cases contributing to the verification of all requirements related to the model element under consideration.

Tables 5 associates the SysML elements with the requirements they satisfied. “Submachine State” and “Atomic State” are the top-level and state machine states, respectively. The “Constraints" are the LTL formulas used to relate the most complex requirements to execution traces as explained in the following paragraphs.

The complexity of «satisfy» relations between structural or behavioural model elements depends on the complexity of the requirement and the way each requirement is reflected by the structural and behavioural model. Consider, for example (see Table 1), requirement

> REQ-3.13.10.2.1: The train speed indicated to the driver shall be identical to the speed used for the speed monitoring (i.e. the estimated speed V_{est}).

Every model trace where the CSM is activated is suitable for verifying this requirement, because the DMI variable speedToDriver is updated by the actual speed V_{est} via operation calc_speed_to_driver(), whenever the ceiling speed monitor is active, that is, in composite state CSM_ON. Therefore CSM_ON is linked to REQ-3.13.10.2.1 by the «satisfy» relation, as expressed in Table 5, row 1.

SysML also allows to express traceability relationships in a graphical way, by drawing arrows from model elements to requirements as shown in Fig. 6. This technique, however, tends to clutter structural and behavioural diagrams as soon as more than a few requirements are involved. Therefore the tabular notation in Table 5 is preferable and supported by most state-of-the-art SysML modelling tools.

A more complex case of requirements tracing presents itself if one or more transitions are related to a given requirement. This is the case for the requirement

> REQ-3.13.10.2.2: Once a Train Interface command (traction cut-off, service brake or emergency brake) is triggered, the on-board shall apply it until its corresponding revocation condition is met.

As modelled in Fig. 5, we have two revocation conditions; one is reflected by the transition from basic state SERVICE_BRAKE to NORMAL, the other from EMER_BRAKE to NORMAL. Therefore both transitions are related to REQ-3.13.10.2.2, as specified in row 2 of Table 5.

Table 5. Requirements links to the SysML Elements

No.	Requirement	⟵ «satisfy»
1	REQ-3.13.10.2.1	«Composite State» CSM_ON
2	REQ-3.13.10.2.2	«Transition» [EMER_BRAKE - NORMAL] «Transition» [SERVICE_BRAKE - NORMAL]
3	REQ-3.13.10.2.3	«Transition» [CSM_OFF - CSM_ON] «Basic State» SERVICE_BRAKE «Constraint» constraint_03
4	REQ-3.13.10.2.4	«Constraint» constraint_02 «Transition» [EMER_BRAKE - NORMAL] «Constraint» constraint_01
5	REQ-3.13.10.2.5	«Transition» [EMER_BRAKE - NORMAL] «Transition» [SERVICE_BRAKE - NORMAL]
6	REQ-3.13.10.3.1	«Submachine State» CSM_ON
7	REQ-3.13.10.3.2	«Basic State» OVERSPEED «Basic State» SERVICE_BRAKE «Basic State» WARNING «Basic State» EMER_BRAKE
8	REQ-3.13.10.3.3.r0	«Transition» [EMER_BRAKE - NORMAL]
9	REQ-3.13.10.3.3.r1	«Transition» [OVERSPEED - NORMAL] «Transition» [SERVICE_BRAKE - NORMAL] «Transition» [WARNING - NORMAL] «Transition» [EMER_BRAKE - NORMAL]
10	REQ-3.13.10.3.3.t1	«Basic State» NORMAL
11	REQ-3.13.10.3.3.t2	«Basic State» OVERSPEED
12	REQ-3.13.10.3.3.t3	«Basic State» WARNING
13	REQ-3.13.10.3.3.t4	«Basic State» SERVICE_BRAKE
14	REQ-3.13.10.3.3.t5	«Basic State» EMER_BRAKE
15	REQ-3.13.10.3.4.r1c3	«Transition» [OVERSPEED - NORMAL]
16	REQ-3.13.10.3.4.r1c4	«Transition» [WARNING - NORMAL]
17	REQ-3.13.10.3.4.r1c5	«Transition» [EMER_BRAKE - NORMAL] «Transition» [SERVICE_BRAKE - NORMAL]
18	REQ-3.13.10.3.4.r3c1	«Transition» [NORMAL - OVERSPEED]
19	REQ-3.13.10.3.4.r4c1	«Constraint» constraint_08
20	REQ-3.13.10.3.4.r4c3	«Transition» [OVERSPEED - WARNING]
21	REQ-3.13.10.3.4.r5c1	«Constraint» constraint_10 «Constraint» constraint_09
22	REQ-3.13.10.3.4.r5c3	«Constraint» constraint_12 «Constraint» constraint_11
23	REQ-3.13.10.3.4.r5c4	«Transition» [WARNING - SERVICE_BRAKE] «Transition» [SERVICE_BRAKE - EMER_BRAKE] «Constraint» constraint_13
24	REQ-3.13.10.3.5	«Constraint» constraint_05 «Constraint» constraint_06 «Constraint» constraint_07 «Basic State» NORMAL «Constraint» constraint_04
25	REQ-3.13.10.3.6	«Constraint» constraint_05 «Constraint» constraint_06 «Constraint» constraint_07 «Constraint» constraint_04

The constraints constraint_01,. . . ,constraint_13 are specified in Table 6.

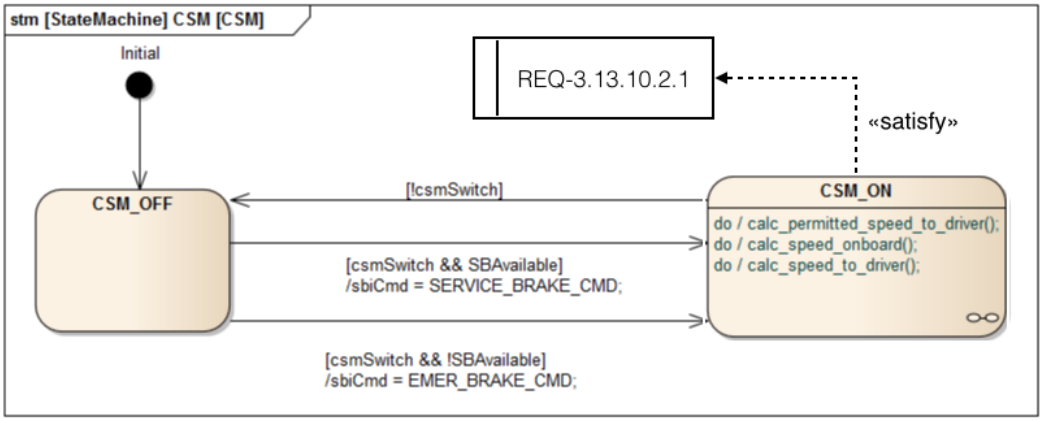


Figure 6. Graphical representation of the «satisfy» relation in a state machine diagram.

In the most complex case we have to handle situations where requirements are reflected by traces visiting model state *vectors*[2] fulfilling certain constraints, and these model state vectors have to be visited by the traces in a specific order. Such a situation is reflected, for example, by

> REQ-3.13.10.3.4: The on-board equipment shall execute the transitions between the different supervision statuses as described in Table 4 (see [28, 4.6.1] for details about the symbols). This table takes into account the order of precedence between the supervision statuses and the possible updates of the MRSP while in ceiling speed monitoring (e.g. when a TSR is revoked; TSR = Temporary Speed Restriction).

This requirement has been decomposed into atomic sub-requirements REQ-3.13.10.3.4.r1c3, . . . , REQ-3.13.10.3.4.r5c4, as explained in Section 3.6. Some of these sub-requirements are again reflected by transitions, as specified in rows 15, 16, 17, 18, and 20 of Table 5. Requirement REQ-3.13.10.3.4.r5c1, however, specifies the possibility to directly transit from NORMAL to SERVICE_BRAKE or EMER_BRAKE. This cannot be specified by simply linking a behavioural model element to the requirement, because we have avoided to draw direct state machine transitions from NORMAL to SERVICE_BRAKE or EMER_BRAKE, since those transitions are implicitly realised by the run-to-completion semantics, as explained above. Consider, for example, the zero-time transition NORMAL ⟶ SERVICE_BRAKE. To cover this situation, we need to

1. Enter NORMAL in a quiescent model state – this is specified by

$$[\text{NORMAL} \wedge V_{est} \leq V_{MRSP}]$$

2. Stay there until the speed exceeds $V_{MRSP} + \text{dV}_{\text{sbi}}(V_{MRSP})$ but remains less or equal to $V_{MRSP} + \text{dV}_{\text{ebi}}$.

[2] A model state vector consists of valuations of inputs, outputs, and internal model variables, as well as of variable valuations indicating the basic state machine states currently active.

Formally this is expressed in LTL by

$$\begin{aligned}&\mathsf{Finally}([\mathsf{NORMAL} \wedge V_{est} \leq V_{MRSP}] \wedge \\ &\quad ([\mathsf{NORMAL} \wedge V_{est} \leq V_{MRSP}] \\ &\quad\ \mathsf{Until} \\ &\quad\ [\mathsf{NORMAL} \wedge V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) \wedge \\ &\quad\ V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})]))\end{aligned}$$

This is expressed by constraint_09 linked to REQ-3.13.10.3.4.r5c1 in Table 5 and specified in Table 6. Similarly, covering the zero-time transition NORMAL $\longrightarrow$ EMER_BRAKE requires a trace satisfying

$$\begin{aligned}&\mathsf{Finally}([\mathsf{NORMAL} \wedge V_{est} \leq V_{MRSP}] \wedge \\ &\quad ([\mathsf{NORMAL} \wedge V_{est} \leq V_{MRSP}] \\ &\quad\ \mathsf{Until} \\ &\quad\ [\mathsf{NORMAL} \wedge V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})]))\end{aligned}$$

(this is specified as constraint_10 in Table 6). Similar constraints are specified for REQ-3.13.10.3.4.r4c1, REQ-3.13.10.3.4.r5c3, and REQ-3.13.10.3.4.r5c4, and for requirements REQ-3.13.10.3.5 and REQ-3.13.10.3.6.

Table 6. Constraints related to complex requirements listed in Table 5.

«Constraint»	**LTL Formula**
constraint_01	**Finally**[EMER_BRAKE $\wedge$ $\neg$SBAvailable $\wedge$ $V_{est} > 0$ $\wedge$ $V_{est} \leq V_{MRSP}$ $\wedge$ $\neg$allowRevokeEB]
constraint_02	**Finally**[SERVICE_BRAKE $\wedge$ $\neg$SBAvailable $\wedge$ $V_{est} \leq V_{MRSP}$]
constraint_03	**Finally**[SERVICE_BRAKE$\wedge\neg$SBAvailable$\wedge V_{est} > V_{MRSP}+\mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})\wedge V_{est} \leq V_{MRSP}+\mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]
constraint_04	**Finally**[CSM_OFF $\wedge$ csmSwitch $\wedge$ $V_{est} > V_{MRSP}$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$]
constraint_05	**Finally**[CSM_OFF $\wedge$ csmSwitch $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$]
constraint_06	**Finally**[CSM_OFF $\wedge$ csmSwitch $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]
constraint_07	**Finally**[CSM_OFF $\wedge$ csmSwitch $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]
constraint_08	**Finally**([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [NORMAL $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$]))
constraint_09	**Finally**([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [NORMAL $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]))
constraint_10	**Finally**([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([NORMAL $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [NORMAL $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]))
constraint_11	**Finally**([OVERSPEED $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([OVERSPEED $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [OVERSPEED $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$ $\wedge$ $V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]))
constraint_12	**Finally**([OVERSPEED $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([OVERSPEED $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [OVERSPEED $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]))
constraint_13	**Finally**([WARNING $\wedge$ $V_{est} \leq V_{MRSP}$] $\wedge$ ([WARNING $\wedge$ $V_{est} \leq V_{MRSP}$] **Until** [WARNING $\wedge$ $V_{est} > V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$]))

4 Formal Semantics – the Transition Relation

4.1 Semantic Definition Scope

In this section the formal behavioural semantics of the CSM model $\mathcal{M}$ written in SysML is specified. The exposition is restricted to the situation where the CSM has already been switched on; this scenario is completely captured by state machine CSM_ON depicted in Fig. 5. This restriction is motivated by the fact that the activation/deactivation scenario for the CSM as shown in state machine CSM (Fig. 4) is incomplete without the models for TSM and RSM. These model components define the SUT behaviour when CSM is in state CSM_OFF.

4.2 State Transition System Semantics

The behaviour of the CSM SysML model $\mathcal{M}$ can be formalised by mapping $\mathcal{M}$ to a *state transition system (STS)* $\mathcal{S} = (S, s_0, R)$ with state space S, initial state $s_0 \in S$ and transition relation $R \subseteq S \times S$. An infinite sequence $\pi = s.s_1.s_2 \ldots$ of $\mathcal{S}$-states is called a *computation* or a *path* of $\mathcal{S}$, if and only if it satisfies

$$s = s_0 \wedge (\forall i \in \mathbb{N} : (s_{i-1}, s_i) \in R)$$

A finite computation prefix is called a *trace*.

The behavioural semantics of a SysML model $\mathcal{M}$ is given by the set of computations that can be executed by the state transition system $\mathcal{S}$ associated with $\mathcal{M}$.

4.3 State Space

For mapping $\mathcal{M}$ to $\mathcal{S}$, we use state spaces over variable valuations: let V be a finite set of variable symbols for variables $v \in V$ with values in some domain $D = \bigcup_{v \in V} D_v$. The state space S of $\mathcal{S}$ is the set of all variable valuations $s : V \to D$ with $s(v) \in D_v$. The variables of V are partitioned into input variables, internal model variables, and output variables; for the CSM model this results in

$$\begin{aligned} V &= I \cup M \cup O \\ I &= \{V_{est}, V_{MRSP}, \text{allowRevokeEB}\} \\ M &= \{\ell, \text{sbiCmd}\} \\ O &= \{\text{DMICmd}, \text{TICmd}\} \end{aligned}$$

These variables have domains as introduced in Section 3, but here we use integer values instead of enumeration types. The enumeration to integer association is defined in Fig. 1.

$$\begin{aligned} &D_{V_{est}} = D_{V_{MRSP}} = [0, 350] \\ &D_{\text{allowRevokeEB}} = D_{\text{sbiCmd}} = D_\ell = \{0, 1\} \\ &D_{\text{TICmd}} = \{0, 1, 2\} \\ &D_{\text{DMICmd}} = \{0, 1, 2, 3, 4\} \end{aligned}$$

Internal variable ℓ does not occur in the SysML model described in Section 3; it is used here to reduce the state space: each of the basic states NORMAL, OVERSPEED, WARNING, as well as the intervention states of the state machine in Fig. 5 are associated with specific outputs on DMICmd. Therefore *all* basic states of this machine are completely identified if we have a means to distinguish SERVICE_BRAKE from EMER_BRAKE in the case DMICmd = 4

(INTERVENTION). To this end, auxiliary variable ℓ will be set to 1 if the state machine of Fig. 5 resides in basic state SERVICE_BRAKE; otherwise ℓ will be set to 0. Using ℓ in this way allows us to identify the basic states as specified in Table 7.

Table 7. Identification of basic states in machine CSM_ON

State Machine in Basic State	Equivalent to
NORMAL	DMICmd = 0
OVERSPEED	DMICmd = 2
WARNING	DMICmd = 3
SERVICE_BRAKE	DMICmd = 4 $\wedge$ ℓ = 1
EMER_BRAKE	DMICmd = 4 $\wedge$ ℓ = 0

We use a short-hand tuple notation for states: $s = (d_1, \ldots, d_7)$ with $d_i \in D$ denotes the state satisfying

$$s(V_{est}) = d_1, s(V_{MRSP}) = d_2, s(\text{allowRevokeEB}) = d_3,$$
$$s(\ell) = d_4, s(\text{sbiCmd}) = d_5,$$
$$s(\text{DMICmd}) = d_6, s(\text{TICmd}) = d_7$$

4.4 Quiescent and Transient States

The STS covered by our testing theory have state spaces that can be partitioned into disjoint sets $S = S_Q \cup S_T$, where S_Q denotes *quiescent* states and S_T *transient* states. In quiescent states, the system is stable and cannot progress without a change of inputs. When these inputs change, internal states and outputs remain unchanged. Transitions emanating from quiescent states may lead to other quiescent states or to transient states. In contrast to this, each transient state has exactly one post-state which is quiescent, and the transition to this post-state is immediate and may change internal state variables and outputs only. The initial state s_0 of the STS under consideration is always quiescent.

From the intuitive interpretation of the CSM model, we see, for example, that the states $s_i = (V_{est}, V_{MRSP}, \text{allowRevokeEB}, \ell, \text{sbiCmd}, \text{DMICmd}, \text{TICmd})$,

$$\begin{aligned} s_0 &= (50, 100, 0, 0, 1, 0, 0) \\ s_1 &= (100.1, 100, 0, 1, 1, 4, 1) \\ s_2 &= (50, 100, 0, 0, 1, 4, 2) \end{aligned}$$

are quiescent, while the states

$$\begin{aligned} s_3 &= (100.7, 100, 0, 0, 1, 0, 0) \\ s_4 &= (99.8, 100, 0, 1, 1, 4, 1) \\ s_5 &= (0, 100, 0, 0, 1, 4, 2) \end{aligned}$$

are transient.

4.5 Initial State

The initial state corresponds to the CSM in its de-activated state. We associate the deactivation state with encoding

$$s_0 = (0, 0, 0, 0, 0, 0, 0)$$

and the first input change activates the CSM.

The existence of a service brake (sbiCmd = 1) is a configuration parameter of the train hardware. Therefore it cannot be regarded as an input to the CSM that may change arbitrarily during its execution, but as a constant value sbiCmd = $\mathrm{sb}_0 \in \{0, 1\}$ that remains invariant during each test execution.

4.6 Transition Relation – General Construction Rules

The transition relation $R \subseteq S \times S$ specifying the possible state changes of the STS associated with the CSM is infinite, since the input domains are infinite. The transition relation, however, can be represented by means of a finite predicate $\mathcal{R}$ relating pre-states to post-states. $\mathcal{R}$ is a proposition with free variables in V and $V' = \{v' \mid v \in V\}$, the primed variable symbols denoting post-states. The transition relation is specified by $\mathcal{R}$ via

$$R = \{(s_0, s_1) \in S \times S \mid \mathcal{R}[s_0(v)/v, s_1(v)/v' \mid v \in V]\}$$

Predicate $\mathcal{R}[s_0(v)/v, s_1(v)/v' \mid v \in V]$ results from $\mathcal{R}$ by replacing every unprimed variable $v \in V$ occurring in R by its pre-state value $s_0(v)$, and every primed variable $v' \in V'$ by its post-state value $s_1(v)$.

The propositional representation of the transition relation is crucial for automated test data generation: as explained in [23], concrete test data is calculated by means of constraint solving techniques applied to formulas of the type

$$tc \equiv J(s_0) \wedge \bigwedge_{i=1}^{n} R(s_{i-1}, s_i) \wedge G(s_0, \ldots, s_n)$$

where $G(s_0, \ldots, s_{n+1})$ encodes a test objective and $\bigwedge_{i=1}^{n} R(s_{i-1}, s_i)$ asserts that every solution is a trace of the model. Each $R(s_{i-1}, s_i)$ is represented by means of $\mathcal{R}$ as a proposition where new inputs to be selected in quiescent state s_i can be freely chosen by the solver, but their effect on internal model variables and outputs is determined by $\mathcal{R}$. Thus we are interested in propositional representations $\mathcal{R}$ mainly for the purpose of tool construction.

While $\mathcal{R}$ is not uniquely determined in the general case, well-defined construction rules are implied by [12], if $\mathcal{R}$ is to be applied for the purpose of equivalence class construction.

1. The canonic structure of $\mathcal{R}$ is

$$\mathcal{R} \equiv \bigvee_{i=0}^{k} (\varphi_i(V) \wedge \psi_i(V, V'))$$

   where $\varphi_i(V)$ is a proposition with free variables in V only, and $\psi_i(V, V')$ has free variables in V and V'. $\varphi_i(V)$ describes the precondition for a transition to fire, and $\psi_i(V, V')$ specifies the transition's effect.

2. Each $\varphi_i(V)$ specifies a set of reachable quiescent states, or a set of reachable transient states.

3. All atomic propositions occurring in $\mathcal{R}$ and involving primed or unprimed internal state variables or output variables v, v', $v \in M \cup O$, must be of the form

$$v = d \text{ or } v' = d, \quad \text{with } d \in D_v$$

Recall that the domains of internal state and outputs variables are finite, therefore the number of atomic propositions $v = d$ is finite. Moreover, every atomic proposition involving variables from I and $M \cup O$ can be transformed into a disjunction of conjunctions of atomic propositions p with free variables from either I or $M \cup O$ only[3].

4. For transitions emanating from quiescent states, $\varphi_i(V)$ is constructed according to the rules

   - Every state s satisfying $\varphi_i(V)$ must be reachable and quiescent.
   - For any reachable quiescent state s there is a unique $\varphi_i(V)$ such that s satisfies $\varphi_i(V)$.
   - Every state s satisfying $\varphi_i(V)$ has the same valuation of internal state and output variables.
   - For any reachable quiescent states s and r with the same valuation of internal state and output variables, s and r satisfy the same $\varphi_i(V)$.

   Again, since the domains of internal state and outputs variables are finite, this construction rule leads to a finite number of propositions $\varphi_i(V)$. We assume that propositions $\varphi_i(V), i = 0, \ldots, k_0 - 1$ specify quiescent states.

5. For transitions emanating from transient states, $\varphi_i(V)$ is constructed according to the rules

   - Every state s satisfying $\varphi_i(V)$ must be reachable and transient.
   - For any transient state s there is a unique $\varphi_i(V)$ such that s satisfies $\varphi_i(V)$.
   - There exists an index j, such that $\varphi_j(V)$ specifies a set of quiescent states according to Rule 3, and every state s satisfying $\varphi_i(V)$ has a post-state satisfying $\varphi_j(V)$.
   - For any transient states s and r with post-states satisfying the same $\varphi_j(V)$, s and r satisfy the same $\varphi_i(V)$.

We assume that $\varphi_{k_0+i}(V), i = 0, \ldots, \leq k_0 - 1$ specify transient states. By construction of $\varphi_i(V)$ and $\varphi_{k_0+i}(V)$, the number of propositions specifying quiescent states according to Rule 4 is greater than or equals the number of transient-state propositions built according to Rule 5. This is easy to see, because each transient state s satisfying $\varphi_{k_0+i}(V)$ has a post-state satisfying the same $\varphi_j(V)$, and this $\varphi_j(V)$ is unique. If the number of quiescent $\varphi_i(V)$ is grater than the number of transient $\varphi_{k_0+i}(V)$, then there are some reachable quiescent states which do not have any transient pre-states. This may only occur for the quiescent state s_0. As a consequence we can assume that proposition $\varphi_0(V)$ specifies the quiescent initial state, and the propositions $\varphi_k(V)$, $k > 0$, are ordered in such a way that the quiescent post-states visited from transient states satisfying $\varphi_{k_0+i}(V)$ are all specified by $\varphi_i(V)$. The existence of $\varphi_{k_0}(V)$ depends on whether there is a transition from transient states to s_0 or if the initial state is never re-visited.

In the case of quiescent pre-states, $\psi_i(V, V')$ is always the same proposition $\text{qpsc}(V, V')$, where "qpsc" stands for *quiescent post-state condition*.

$$\text{qpsc}(V, V') \equiv \left(\bigvee_{x \in I} x' \neq x \right) \wedge \left(\bigwedge_{v \in M \cup O} v' = v \right)$$

[3]Consider, for example, atomic proposition $x < m$ with $D_x = \mathbb{R}$ and $D_m = \{0, 1\}$. Then $x < m \equiv (x < 0 \wedge m = 0) \vee (x < 1 \wedge m = 1)$.

This specifies that at least one input variable must change its value during a transition from a quiescent state, and all internal state and output valuations remain unchanged.

Transitions from transient states to quiescent states always leave inputs unchanged; this leads to the complementary *transient post-state condition*

$$\mathrm{tpsc}(V, V') \quad \equiv \quad \bigwedge_{v \in I} v' = v$$

so the effect of the transition from transient pre-state to quiescent post-state can be written as

$$\psi_{k_0+i}(V, V') \equiv \overline{\psi}_{k_0+i}(V, V') \wedge \mathrm{tpsc}(V, V')$$

where $\overline{\psi}_{k_0+i}(V, V')$ is still to be determined.

As a consequence of Rule 4, every $\varphi_i(V)$ characterising a subset of quiescent states is of the form

$$\varphi_i(V) \quad \equiv \quad \varphi_i^I(I) \wedge \xi_i(M \cup O) \tag{5}$$

$$\xi_i(M \cup O) \quad \equiv \quad \bigwedge_{m \in M} (m = d_i^m) \wedge \bigwedge_{y \in O} (y = d_i^y), \quad d_i^m \in D_m,\ d_i^y \in D_y \tag{6}$$

so that $\varphi_i^I(I)$ is a (generally non-atomic) proposition over free variables from I.

As a consequence of Rule 5 and of our numbering conventions, each $\varphi_{k_0+i}(V)$ characterising a subset of transient states satisfies

$$\forall s_1, s_2 \in S : s_1(\varphi_{k_0+i}) \wedge R(s_1, s_2) \Rightarrow s_2(\varphi_i)$$

This means that $\varphi_{k_0+i}(V)$ is constructed for transient states in such a way that *all* post-states of the transient state set

$$B_i = \{s \in S \mid s(\varphi_{k_0+i}(V))\}$$

are members of the same set

$$A_i = \{s \in S \mid s(\varphi_i(V))\}$$

of quiescent states. Therefore each $\psi_{k_0+i}(V, V')$ specifying the effect of a transition from transient to quiescent state can be structured as

$$\psi_{k_0+i}(V, V') \equiv \xi_i[m'/m, y'/y \mid m \in M, y \in O] \wedge \mathrm{tpsc}(V, V')$$

where $\xi_i[m'/m, y'/y \mid m \in m, y \in O]$ is the proposition specifying internal state and output values for the associated set A_i of quiescent states, but each internal model state variable m and each output variable y are replaced by their primed versions m' and y', respectively. Conversely, if B_i can be reached from quiescent state set A_k by means of input changes satisfying a certain proposition $\varphi_{k,i}^I$, these reachable elements $s \in B_i$ satisfy $\varphi_{k,i}^I \wedge \xi_k$, since internal state values and outputs do not change when transiting from an A_k-state to a B_i-state. As a consequence, the proposition $\varphi_{k_0+i}(V)$ characterising B_i-states has a canonic structure

$$\varphi_{k_0+i}(V) \equiv \bigvee_{q=0}^{k_0-1} (\varphi_{q,i}^I(V) \wedge \xi_q(V)) \tag{7}$$

where $\varphi_{q,i}^I(V) \equiv \texttt{false}$, if there are no transitions from A_q-states to B_i-states.

Summarising, the systems covered by our equivalence class testing theory have a behavioural semantics that can be expressed by an associated STS whose transition relation can be specified by means of a proposition

$$\mathcal{R}(V,V') \equiv \bigvee_{i=0}^{k_0-1} (\varphi_i^I(I) \wedge \xi_i(M \cup O) \wedge \mathrm{qpsc}(V,V')) \vee \tag{8}$$

$$\bigvee_{i=0}^{k_0-1} ((\bigvee_{q=0}^{k_0-1} \varphi_{q,i}^I(I) \wedge \xi_q(M \cup O)) \wedge \tag{9}$$

$$\xi_i[m'/m, y', y \mid m \in M, y \in O] \wedge \mathrm{tpsc}(V,V')) \tag{10}$$

where $k_0 + 1$ is the number of reachable quiescent state classes A_i, each determined by a specific valuation of internal states and outputs, as specified by ξ_i.

4.7 Transition Relation for the CSM

With the preparations of the previous section at hand, we are now in the position to specify the proposition $\mathcal{R}$ for the ceiling speed monitor, as applicable to all states of S that are reachable from initial state s_0. To this end, the propositions occurring in the canonic representation shown in Equation (8) are specified one by one.

4.7.1 Propositions Specifying Internal State and Outputs – ξ_i.

The following combinations of internal state values and output values are reachable; this results in $k_0 = 5$ and in the following propositions.

$$\begin{aligned}
\xi_0(V) &\equiv \ell = 0 \wedge \mathrm{DMICmd} = 0 \wedge \mathrm{TICmd} = 0\\
\xi_1(V) &\equiv \ell = 0 \wedge \mathrm{DMICmd} = 2 \wedge \mathrm{TICmd} = 0\\
\xi_2(V) &\equiv \ell = 0 \wedge \mathrm{DMICmd} = 3 \wedge \mathrm{TICmd} = 0\\
\xi_3(V) &\equiv \ell = 1 \wedge \mathrm{DMICmd} = 4 \wedge \mathrm{TICmd} = 2 - \mathrm{sb}_0\\
\xi_4(V) &\equiv \ell = 0 \wedge \mathrm{DMICmd} = 4 \wedge \mathrm{TICmd} = 2
\end{aligned}$$

4.7.2 Propositions Specifying Input Conditions for Quiescent Classes – φ_i^I.

The input conditions for the 5 quiescent class $A_0, \ldots, A_4$ – each class A_i associated with internal state and output valuation ξ_i – are defined as follows.

$$\varphi_0^I(V) \equiv V_{est} \leq V_{MRSP} \tag{11}$$

$$\varphi_1^I(V) \equiv V_{MRSP} < V_{est} \wedge V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP}) \tag{12}$$

$$\varphi_2^I(V) \equiv V_{MRSP} < V_{est} \wedge V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) \tag{13}$$

$$\varphi_3^I(V) \equiv V_{MRSP} < V_{est} \wedge V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP}) \tag{14}$$

$$\varphi_4^I(V) \equiv (0 < V_{est} \wedge \mathrm{allowRevokeEB} = 0) \vee (V_{MRSP} < V_{est} \wedge \mathrm{allowRevokeEB} = 1) \tag{15}$$

The propositions $\varphi_i^I(V), \xi_i(V)$ cover the following quiescent state classes.

$\varphi_0^I(V) \wedge \xi_0(V)$ specifies the quiescent states associated with basic state NORMAL, while the train speed does not exceed V_{MRSP}.

$\varphi_1^I(V) \wedge \xi_1(V)$ specifies the quiescent states associated with basic state OVERSPEED, while the train speed does not exceed the boundary for transiting to the WARNING state, and the speed has not yet been reduced enough to transit back to NORMAL.

$\varphi_2^I(V) \wedge \xi_2(V)$ specifies the quiescent states associated with basic state WARNING, while the train speed does not exceed the boundary for transiting to the SERVICE_BRAKE state, and the speed has not yet been reduced enough to transit back to NORMAL.

$\varphi_3^I(V) \wedge \xi_3(V)$ specifies the quiescent states associated with basic state SERVICE_BRAKE, while the train speed does not exceed the boundary for transiting to the EMER_BRAKE state, and the speed has not yet been reduced enough to transit back to NORMAL. Output specification TICmd = $2-\mathrm{sb}_0$ which is part of $\xi_3(V)$ requires that the service brake is triggered (TICmd = 1) if such a brake is available (constant $\mathrm{sb}_0 = 1$); otherwise the emergency brake is triggered (TICmd = 2).

$\varphi_4^I(V) \wedge \xi_4(V)$ specifies the quiescent states associated with basic state EMER_BRAKE, while either

- the speed V_{est} is still greater zero and input allowRevokeEB = 0 forbids to return to NORMAL before the train has come to a standstill, or
- the CSM may return to NORMAL as soon as $V_{est} \leq V_{MRSP}$ because allowRevokeEB = 1, but currently the estimated speed is still greater than V_{MRSP}.

For building more fine-grained equivalence classes (see Section 5 below) it is important to observe that the functions $\mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$, $\mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$, $\mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$ contain internal case distinctions as specified in Equations (2 — 4). Inserting these case distinctions into the inequalities referencing these functions in propositions $\varphi_i^I(V)$, $i = 1, 2, 3$ results in refined predicates

$$\varphi_1^I(V) \equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 4) \vee \quad \textbf{(16)}$$

$$(110 < V_{MRSP} \leq 140 \wedge V_{MRSP} < V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3}) \vee \quad \textbf{(17)}$$

$$(140 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5) \quad \textbf{(18)}$$

$$\varphi_2^I(V) \equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5.5) \vee \quad \textbf{(19)}$$

$$(110 < V_{MRSP} \leq 210 \wedge V_{MRSP} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100}) \vee \quad \textbf{(20)}$$

$$(210 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 10) \quad \textbf{(21)}$$

$$\varphi_3^I(V) \equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 7.5) \vee \quad \textbf{(22)}$$

$$(110 < V_{MRSP} \leq 210 \wedge V_{MRSP} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4}) \vee \quad \textbf{(23)}$$

$$(210 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 15) \quad \textbf{(24)}$$

4.7.3 Quiescent Post-State Condition – qpsc.

For the CSM, this conditions is defined as follows.

$$\begin{aligned} \mathrm{qpsc} \equiv\ & (V_{est}', V_{MRSP}', \mathrm{allowRevokeEB}') \neq (V_{est}, V_{MRSP}, \mathrm{allowRevokeEB}) \wedge \\ & \ell' = \ell \wedge \mathrm{DMICmd}' = \mathrm{DMICmd} \wedge \mathrm{TICmd}' = \mathrm{TICmd} \wedge \mathrm{sbiCmd}' = \mathrm{sb}_0 \end{aligned}$$

4.7.4 Transient Post-State Condition – tpsc.

For the CSM, this conditions is defined as follows.

$$\begin{aligned} \mathrm{tpsc} \equiv\ & V_{est}' = V_{est} \wedge V_{MRSP}' = V_{MRSP} \wedge \mathrm{allowRevokeEB}' = \mathrm{allowRevokeEB} \wedge \\ & \mathrm{sbiCmd}' = \mathrm{sb}_0 \end{aligned}$$

4.7.5 Transient State Input Conditions – $\varphi_{q,i}^I$.

As explained in the previous section, the transient states of some class B_i are characterised by disjunctions of predicates $\varphi_{q,i}^I(V) \wedge \xi_q(V)$, where $\varphi_{q,i}^I$ denotes the condition on input changes

required to reach B_i states from A_q states. These propositions are specified as follows.

$$\varphi^I_{0,1}(V) \equiv V_{MRSP} < V_{est} \wedge V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP}) \quad (25)$$
$$\varphi^I_{0,2}(V) \equiv V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP}) < V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) \quad (26)$$
$$\varphi^I_{0,3}(V) \equiv V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) < V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{ebi}}(V_{MRSP}) \quad (27)$$
$$\varphi^I_{0,4}(V) \equiv V_{MRSP} + \mathrm{dV}_{\mathsf{sbi}}(V_{MRSP}) < V_{est} \quad (28)$$
$$\varphi^I_{1,2}(V) \equiv \varphi^I_{0,2}(V) \quad (29)$$
$$\varphi^I_{1,3}(V) \equiv \varphi^I_{0,3}(V) \quad (30)$$
$$\varphi^I_{1,4}(V) \equiv \varphi^I_{0,4}(V) \quad (31)$$
$$\varphi^I_{2,3}(V) \equiv \varphi^I_{0,3}(V) \quad (32)$$
$$\varphi^I_{2,4}(V) \equiv \varphi^I_{0,4}(V) \quad (33)$$
$$\varphi^I_{3,4}(V) \equiv \varphi^I_{0,4}(V) \quad (34)$$
$$\varphi^I_{1,0}(V) \equiv V_{est} \leq V_{MRSP} \quad (35)$$
$$\varphi^I_{2,0}(V) \equiv \varphi^I_{1,0}(V) \quad (36)$$
$$\varphi^I_{3,0}(V) \equiv \varphi^I_{1,0}(V) \quad (37)$$
$$\varphi^I_{4,0}(V) \equiv V_{est} = 0 \vee (V_{est} \leq V_{MRSP} \wedge \mathrm{allowRevokeEB} = 1) \quad (38)$$
$$(39)$$

Again, taking into account the internal decisions in $\mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$, $\mathrm{dV}_{\mathsf{sbi}}(V_{MRSP})$, and $\mathrm{dV}_{\mathsf{ebi}}(V_{MRSP})$, the propositions representing these functions can be refined; this leads to

$$\begin{aligned}
\varphi^I_{0,1}(V) \equiv\ & V_{MRSP} < V_{est} \wedge \\
& ((V_{MRSP} \leq 110 \wedge V_{est} \leq V_{MRSP} + 4) \vee \\
& (110 < V_{MRSP} \leq 140 \wedge V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3}) \vee \\
& (140 < V_{MRSP} \wedge V_{est} \leq V_{MRSP} + 5)) \\
\varphi^I_{0,2}(V) \equiv\ & (V_{MRSP} \leq 110 \wedge V_{MRSP} + 4 < V_{est} \leq V_{MRSP} + 5.5) \vee \\
& (110 < V_{MRSP} \leq 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100}) \vee \\
& (140 < V_{MRSP} \leq 210 \wedge V_{MRSP} + 5 < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100}) \vee \\
& (210 < V_{MRSP} \wedge V_{MRSP} + 5 < V_{est} \leq V_{MRSP} + 10) \\
\varphi^I_{0,3}(V) \equiv\ & (V_{MRSP} \leq 110 \wedge V_{MRSP} + 5.5 < V_{est} \leq V_{MRSP} + 7.5) \vee \\
& (110 < V_{MRSP} \leq 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4}) \vee \\
& (210 < V_{MRSP} \wedge V_{MRSP} + 10 < V_{est} \leq V_{MRSP} + 15) \\
\varphi^I_{0,4}(V) \equiv\ & (V_{MRSP} \leq 110 \wedge V_{MRSP} + 7.5 < V_{est}) \vee \\
& (110 < V_{MRSP} \leq 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est}) \vee \\
& (210 < V_{MRSP} \wedge V_{MRSP} + 15 < V_{est})
\end{aligned}$$

Summarising, the transition relation of the CSM is specified by the following proposition, where ϕ_i has been introduced in Equation (5), ϕ_{k_0+i} in Equation (7), and all terms φ^I_i, qpsc, tpsc, $\varphi^I_{q,i}$,

and ξ_i have been specified above.

$$\mathcal{R} \equiv \bigvee_{i=0}^{k_0-1} (\varphi_i \wedge \text{qpsc}) \vee \bigvee_{i=0}^{k_0-1} (\varphi_{k_0+i} \wedge \xi_i[m'/m, y'/y \mid m \in M, y \in O] \wedge \text{tpsc}) \quad \textbf{(40)}$$

$$k_0 = 5 \quad \textbf{(41)}$$

$$\varphi_0 = \varphi_0^I \wedge \xi_0 \quad \textbf{(42)}$$

$$\varphi_1 = \varphi_1^I \wedge \xi_1 \quad \textbf{(43)}$$

$$\varphi_2 = \varphi_2^I \wedge \xi_2 \quad \textbf{(44)}$$

$$\varphi_3 = \varphi_3^I \wedge \xi_3 \quad \textbf{(45)}$$

$$\varphi_4 = \varphi_4^I \wedge \xi_4 \quad \textbf{(46)}$$

$$\varphi_{k_0} = \bigvee_{q=1}^{4} (\varphi_{q,0}^I \wedge \xi_q) \quad \textbf{(47)}$$

$$\varphi_{k_0+1} = (\varphi_{0,1}^I \wedge \xi_0) \quad \textbf{(48)}$$

$$\varphi_{k_0+2} = (\varphi_{0,2}^I \wedge (\xi_0 \vee \xi_1)) \quad \textbf{(49)}$$

$$\varphi_{k_0+3} = (\varphi_{0,3}^I \wedge (\xi_0 \vee \xi_1 \vee \xi_2)) \quad \textbf{(50)}$$

$$\varphi_{k_0+4} = (\varphi_{0,4}^I \wedge (\xi_0 \vee \xi_1 \vee \xi_2 \vee \xi_3)) \quad \textbf{(51)}$$

5 Input Equivalence Class Partitionings

5.1 Strategy Overview

In this section we summarise the main results of the novel equivalence class partitioning method, whose theory has been described in [12], before its application is illustrated in Section 5.2, using the CSM as an example.

In the exposition below, variable symbols x, m, y are used with the convention that $x \in I, m \in M, y \in O$, and the symbols can be enumerated as $I = \{x_1, \ldots, x_k\}$, $M = \{m_1, \ldots, m_p\}$, $O = \{y_1, \ldots, y_q\}$. We use notation $\vec{x} = (x_1, \ldots, x_k)$, $s(\vec{x}) = (s(x_1), \ldots, s(x_k))$, $D_I = D_{x_1} \times \cdots \times D_{x_k}$ denotes the cartesian product of the input variable domains. Tuples $\vec{m}, \vec{y}$ and D_M and D_O are defined over model variables and outputs in an analogous way. By $s \oplus \{\vec{x} \mapsto \vec{c}\}$, $\vec{c} \in D_I$ we denote the state s' which coincides with s on all variables from $M \cup O$, but returns values $s'(x_i) = c_i,\ i = 1, \ldots, k$ for the input symbols.

5.1.1 I/O-Equivalence

Applying a trace $\iota = \vec{c}_1 \ldots \vec{c}_n$ of input vectors $\vec{c}_i \in D_I$ to a STS (S, s_0, R) residing in some quiescent state $s \in S$, this stimulates a sequence of state transitions with associated output changes as triggered by the inputs. Restricting this sequence to quiescent states, this results in a trace of states $\tau = s_1.s_2 \ldots s_n$ such that $s_i(\vec{x}) = \vec{c}_i, i = 1, \ldots, n$, and $s_i(\vec{y})$ is the last STS output resulting from application of $\vec{c}_1 \ldots \vec{c}_i$ to state s. This trace τ is generally denoted by s/ι. The restriction of s/ι to output variables is denoted by $(s/\iota)|_O$. Since transient states have unique quiescent post-states, the restriction to quiescent states does not result in a loss of information, if the input trace ι is known: the omitted transient states are some elements of $s \oplus \{\vec{x} \mapsto \vec{c}_1\}, \ldots, s_{n-1} \oplus \{\vec{x} \mapsto \vec{c}_n\}$, and these states satisfy $R(s \oplus \{\vec{x} \mapsto \vec{c}_1\}, s_1), \ldots, R(s_{n-1} \oplus \{\vec{x} \mapsto \vec{c}_n\}, s_n)$.

Two states s, s' are *I/O-equivalent*, written $s \sim s'$, if every non-empty input trace ι, when applied to s and s', results in the same outputs, that is, $(s/\iota)|_O = (s'/\iota)|_O$. Two STS $\mathcal{S}, \mathcal{S}'$ are I/O-equivalent, if their initial states are I/O-equivalent. Note that for technical reasons, $s \sim s'$ still admits that $s|_O \neq s'|_O$.

5.1.2 Input Equivalence Class Partitions

Since I/O-equivalence is an equivalence relation, we can factorise STS state spaces by $\sim$, and the resulting equivalence classes $A \in S/_\sim$ have the property that all $s, s' \in A$ yield the same output traces $(s/\iota)|_O = (s'/\iota)|_O$ for arbitrary non-empty input traces ι. For systems like the CSM, the number of classes A is finite, so we can enumerate $S/_\sim = \{A_1, \ldots, A_r\}$. Applying an arbitrary input vector $\vec{c} \in D_I$ to any state $s \in A_i$ will always lead to a quiescent target state – denoted by $(s//\vec{c})$ – in the same target class A_j. Index j only depends on $(i, \vec{c})$, since for $s, s' \in A_i$ all corresponding states s_k, s'_k in $s/\iota = s_1.s_2 \ldots .s_n, s'/\iota = s'_1.s'_2 \ldots .s'_n$ are I/O-equivalent for any $\iota = \vec{c}_1 \ldots \vec{c}_n, k = 1 \ldots n$.

Therefore $(s//\vec{c}) \in A_j$ if and only if $(s'//\vec{c}) \in A_j$. One class A_j, however, may contain elements $s \sim s'$ with different outputs, since I/O-equivalence only states that all future outputs will be identical, when applying the same non-empty input trace to s, s'. Since $D_O = \{\vec{d}_1, \ldots, \vec{d}_{|D_O|}\}$ is finite, we can associate the value index $h \in \{1, \ldots, |D_O|\}$ with the target class A_j, if $(s//\vec{c})|_O = \vec{d}_h$. Again, h only depends on $(i, \vec{c})$, but not on the choice of $s \in A_i$.

Applying $\vec{c}$ to elements from all classes $A_1, \ldots, A_r$, results in (not necessarily distinct) index pairs $j(\vec{c}, i), h(\vec{c}, i),\ i = 1, \ldots, r$. This induces a factorisation of the input domain D_I: define $X(\vec{c}) \subseteq D_I$ as the maximal set containing $\vec{c}$, such that $j(\vec{c}', i) = j(\vec{c}, i) \wedge h(\vec{c}', i) = h(\vec{c}, i),\ i = 1, \ldots, r$, holds for all $\vec{c}' \in X(\vec{c})$.

Then the *Input Equivalence Class Partitioning (IECP)* $\mathcal{I} = \{X(\vec{c}) \mid \vec{c} \in D_I\}$ has the following properties: (1) The elements of $\mathcal{I}$ are pairwise disjoint, (2) The union of all $X \in \mathcal{I}$ equals D_I, (3) $\mathcal{I}$ is finite, and (4) for all $s \in A_i$, $\vec{c} \in X$, target states $(s//\vec{c})$ are contained in the same target class $A_{j(i,\vec{c})}$ and have the same output value $d_{h(i,\vec{c})}$. Furthermore, each pair of input traces $\iota = \vec{c}_1 \ldots \vec{c}_n$, $\iota' = \vec{c}'_1 \ldots \vec{c}'_n$, when applied to the same state s, lead to the same output traces $(s/\iota)|_O = (s/\iota')|_O$, if $\vec{c}'_i \in X(\vec{c}_i)$ for each $i = 1, \ldots, n$.

A given IECP $\mathcal{I}$ can be *refined* by selecting input sets $\mathcal{I}_2 = \{X_1, X_2, \ldots\}$ such that $\mathcal{I}_2$ also fulfils the above properties (1), (2), (3), and such that every X_i is a subset of some $X \in \mathcal{I}$. If these conditions hold, $\mathcal{I}_2$ inherits property (4). Refinement is obviously reflexive, transitive and anti-symmetric.

5.1.3 Fault Model

As reference models we use the STS representations $\mathcal{S}$ of models elaborated in concrete formalisms – like the CSM model presented in this paper – such that the expected behaviour of the SUT is specified by $\mathcal{S}$ up to I/O-equivalence. We use I/O-equivalence as conformance relation. The fault domain $\mathcal{D}$ specifies the set of potential systems under test, whose true behaviour can be represented by an STS $\mathcal{S}' \in \mathcal{D}$. For the equivalence class testing strategy, the fault domain depends on the reference model $\mathcal{S}$ and two additional parameters $m \in \mathbb{N}$ and a refinement $\mathcal{I}_2$ of $\mathcal{I}$, the IECP associated with $\mathcal{S}$. $\mathcal{D}(\mathcal{S}, m, \mathcal{I}_2)$ contains all $\mathcal{S}'$ satisfying

1. The states of $\mathcal{S}'$ are defined over the same variable space $V = I \cup M \cup O$ as defined for the model $\mathcal{S}$.
2. Initial state s'_0 of $\mathcal{S}'$ coincides with initial state s_0 of $\mathcal{S}$ on $I \cup O$.
3. $\mathcal{S}'$ generates only finitely many different output values and internal state values.
4. The number of I/O-equivalence classes of $\mathcal{S}'$ is less or equal m.
5. Let $\mathcal{I}'$ be the IECP of $\mathcal{S}'$ as defined above. Then
$$\forall X \in \mathcal{I}, X' \in \mathcal{I}' : (X \cap X' \neq \varnothing \Rightarrow \exists X_2 \in \mathcal{I}_2 : X_2 \subseteq X \cap X')$$
6. $\mathcal{S}'$ has a well-defined reset operation allowing to re-start the system, in order to perform another test from its initial state.

Requirement 2 is well-founded, since initial states correspond to the system's switched-off state. Therefore we can assume that the implementation produces the same outputs as the reference model as long as it is switched off – otherwise we would not start testing, because $\mathcal{S}$ and $\mathcal{S}'$ differed already in the off-state.

The fault domain $\mathcal{D}(\mathcal{S}, m, \mathcal{I}_2)$ is obviously increased by increasing $m \in \mathbb{N}$, and/or further refining $\mathcal{I}_2$:
$$m' \geq m \wedge \mathcal{I}_3 \text{ refines } \mathcal{I}_2 \Rightarrow \mathcal{D}(\mathcal{S}, m, \mathcal{I}_2) \subseteq \mathcal{D}(\mathcal{S}, m', \mathcal{I}_3)$$

5.1.4 Complete Test Strategy

The main result of the paper [12] states that, given reference model $\mathcal{S}$ and fixing $(m, \mathcal{I}_2)$, it is possible to generate a finite test suite from $\mathcal{S}$, such that (a) this suite accepts every member of $\mathcal{D}(\mathcal{S}, m, \mathcal{I}_2)$ which is I/O-equivalent to $\mathcal{S}$, and (b) at least one test of this suite fails for every non-conforming member of $\mathcal{D}(\mathcal{S}, m, \mathcal{I}_2)$ which violates the I/O-equivalence condition. Test suites satisfying (a) are called *sound*, and those satisfying (b) are called *exhaustive*. Soundness and exhaustiveness together is called *complete*. The test suite is generated as follows.

1. Select one representative input vector $\vec{c}(X)$ from each $X \in \mathcal{I}_2$.
2. Abstract $\mathcal{S}$ to a finite deterministic state machine $\mathcal{M}$ with I/O-equivalence classes $A_1, \ldots, A_r$ as states, input alphabet $\{\vec{c}(X) \mid X \in \mathcal{I}_2\}$ and output alphabet D_O (recall that D_O is finite). This DFSM is well-defined due to the properties of the $A \in S/_\sim$ and the $X \in \mathcal{I}_2$.
3. Since $\mathcal{M}$ is a DFSM, the well known W-Method [29, 4] can be used to create a test suite that is complete with respect to reference model $\mathcal{M}$, conformance relation DFSM-equivalence, and the set of all DFSM over the same input/output alphabets as fault domain, whose numbers of states do not exceed m.
4. A STS $\mathcal{S}'$ is I/O-equivalent to $\mathcal{S}$ if and only if its DFSM $\mathcal{M}'$ passes these tests, so that $\mathcal{M}'$ is DFSM-equivalent to $\mathcal{M}$.

5.2 Practical Construction of Input Equivalence Classes and Associated Partitionings

The set-theoretic introduction of IECP in Section 5.1 have a propositional counterpart in the formulas introduced above in the context of the transition relation. This propositional view is needed for being able to calculate concrete representatives of IECP with the help of an constraint solver.

5.2.1 CSM I/O-Equivalence Classes

Reviewing the CSM transition relation shown in Formulas (40 — 51), the identification of quiescent I/O equivalence classes is straightforward: they are identical to or unions of the quiescent state sets

$$A_i = \{s \in S \mid s(\varphi_i)\}, \quad i = 0, \ldots, 4, \quad \varphi_i \text{ as defined in formulas (42 — 46)} \tag{52}$$

To see this, let us first show that each A_i only contains I/O-equivalent states, so that it is a subset of an I/O equivalence class. This follows directly from the following observations about the CSM transition relation.

1. If one quiescent state s from A_i can transit to a transient state s' in set

   $$B_j = \{s \mid s(\varphi_{k_0+j})\}, \quad j = 0, \ldots, 4$$

   by an input change $s' = s \oplus \{(V_{est}, V_{MRSP}, \text{allowRevokeEB}) \mapsto (c_1, c_2, c_3)\}$ then *all* elements of A_i can transit to B_j with the same input change $(V_{est}, V_{MRSP}, \text{allowRevokeEB}) \mapsto (c_1, c_2, c_3)$.
2. All elements of B_j have post-states in the same quiescent state set A_j, uniquely determined by the fact that these post-states must satisfy ξ_j, that is, must have the same outputs and the same internal state.

From the calculations performed for the CSM transition relation (Section 4.7.5) we know that $\varphi^I_{0,i} \equiv \varphi^I_{1,i}, i = 2, 3, 4$, $\varphi_0 \equiv \varphi^I_{1,0}$, and $\varphi^I_{0,1} \equiv \varphi_1$. This implies that A_0 and A_1 are I/O-equivalent. Similarly, it can be deduced that $A_0 \cup A_1, A_2, A_3, A_4$ are pairwise distinguishable with respect to I/O-equivalence, so these four sets are the I/O-equivalence classes of the CSM's STS.

5.2.2 CSM Input Equivalence Class Partitions

As described in Section 5.1.2, we need the refined quiescent state sets for the identification of IECP, resulting from I/O-equivalence classes further partitioned in a way that the states of each resulting set have identical output valuations. These are exactly the sets $A_0, \ldots, A_4$ from equation (52), since the formulas φ_i defining the A_i specify the constant output values.

The input equivalence class partitions introduced in Section 5.1.2 can be expressed in a propositional way by specifying

$$\mathcal{I} = \{X = \{\vec{c} \in D_i \mid \bigwedge_{i=0}^{4} \varphi^I_{q,j_i}[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]\} \mid j_i \in \{0, \ldots, 4\} \wedge X \neq \emptyset\} \quad \textbf{(53)}$$

In this definition the φ^I_{q,j_i} are defined as in Equation (25 — 38), for $q \neq j_i$, and $\varphi^I_{q,q} \equiv \varphi^I_q$, with φ^I_q defined in Equation (11 — 15).

For the CSM, the feasible non-equivalent propositions $\Phi_k \equiv \bigwedge_{i=0}^{4} \varphi^I_{q,j_i}$ are

$$\begin{aligned}
\Phi_1 &\equiv \varphi^I_{0,0} \wedge \varphi^I_{1,0} \wedge \varphi^I_{2,0} \wedge \varphi^I_{3,0} \wedge \varphi^I_{4,4} \\
&\equiv \varphi^I_{0,0} \wedge \varphi^I_{4,4} \\
&\equiv 0 < V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 0 \\
\Phi_2 &\equiv \varphi^I_{0,0} \wedge \varphi^I_{1,0} \wedge \varphi^I_{2,0} \wedge \varphi^I_{3,0} \wedge \varphi^I_{4,0} \\
&\equiv \varphi^I_{4,0} \\
&\equiv V_{est} = 0 \vee (V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 1) \\
\Phi_3 &\equiv \varphi^I_{0,1} \wedge \varphi^I_{1,1} \wedge \varphi^I_{2,2} \wedge \varphi^I_{3,3} \wedge \varphi^I_{4,4} \\
&\equiv \varphi^I_{0,1} \wedge \varphi^I_{4,4} \\
&\equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 4) \vee \\
&\quad (110 < V_{MRSP} \leq 140 \wedge V_{MRSP} < V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3}) \vee \\
&\quad (140 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5) \\
\Phi_4 &\equiv \varphi^I_{0,2} \wedge \varphi^I_{1,2} \wedge \varphi^I_{2,2} \wedge \varphi^I_{3,3} \wedge \varphi^I_{4,4} \\
&\equiv \varphi^I_{0,2} \wedge \varphi^I_{4,4} \\
&\equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} + 4 < V_{est} \leq V_{MRSP} + 5.5) \vee \\
&\quad (110 < V_{MRSP} \leq 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100}) \vee \\
&\quad (140 < V_{MRSP} \leq 210 \wedge V_{MRSP} + 5 < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100}) \vee \\
&\quad (210 < V_{MRSP} \wedge V_{MRSP} + 5 < V_{est} \leq V_{MRSP} + 10)
\end{aligned}$$

$$
\begin{aligned}
\Phi_5 &\equiv \varphi^I_{0,3} \wedge \varphi^I_{1,3} \wedge \varphi^I_{2,3} \wedge \varphi^I_{3,3} \wedge \varphi^I_{4,4} \\
&\equiv \varphi^I_{0,3} \\
&\equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} + 5.5 < V_{est} \leq V_{MRSP} + 7.5) \vee \\
&\quad (110 < V_{MRSP} \leq 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4}) \vee \\
&\quad (210 < V_{MRSP} \wedge V_{MRSP} + 10 < V_{est} \leq V_{MRSP} + 15) \\
\Phi_6 &\equiv \varphi^I_{0,4} \wedge \varphi^I_{1,4} \wedge \varphi^I_{2,4} \wedge \varphi^I_{3,4} \wedge \varphi^I_{4,4} \wedge \\
&\equiv \varphi^I_{0,4} \\
&\equiv (V_{MRSP} \leq 110 \wedge V_{MRSP} + 7.5 < V_{est}) \vee \\
&\quad (110 < V_{MRSP} \leq 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est}) \vee \\
&\quad (210 < V_{MRSP} \wedge V_{MRSP} + 15 < V_{est})
\end{aligned}
$$

With these definitions, the IECP is given by

$$\mathcal{I} = \{X_1, \ldots, X_6\} \quad \textbf{(54)}$$

$$X_i = \{\vec{c} \in D_I \mid \Phi_i[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]\}, \quad i = 1, \ldots, 6 \quad \textbf{(55)}$$

The properties of this IECP will be discussed in the subsequent sections, and we will also discuss the necessity for IECP refinements.

5.3 Inter-Class Transitions

With the IECP at hand, the state transition system representing the behavioural semantics of the SysML state machine can be abstracted to a deterministic finite state machine (DFSM): Each X_i introduced above, or a representative value of each X_i, represents an event of the state machine, and the classes A_i, B_i are the DFSM states. The DFSM transition table is specified in Table 8.

For concrete tests we select representatives of each IEC, such as the ones specified in Table 9. The set of these representatives is called the *input alphabet* $\mathcal{A}_I$.

6 CSM Fault Model

The fault model introduced in Section 5.1.3 is instantiated for the CSM as follows.

- The reference model for the CSM is the STS $\mathcal{S}$ defined in Section 4.
- We use I/O-equivalence as conformance relation, as introduced in Section 5.1.1.
- We will consider several fault domains, starting with

  $$\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$$

  where m denotes the upper bound of I/O-equivalence classes occurring in (conforming or non-conforming) implementations of $\mathcal{S}$, and $\mathcal{I}$ is the IECP constructed in Section 5.2.2.

The assurance gained from performing a completed test suite with respect to fault domain $\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$ will be discussed below in Section 8, and meaningful refinements of $\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$ are introduced in Section 9.

Table 8. DFSM Transition Table.

Source	**Input**	**via**	**Target**	DMICmd	TICmd
A_0	$X_1 \cup X_2$	—	A_0	0	0
A_0	X_3	B_1	A_1	2	0
A_0	X_4	B_2	A_2	3	0
A_0	X_5	B_3	A_3	4	2-sb$_0$
A_0	X_6	B_4	A_4	4	2
A_1	$X_1 \cup X_2$	B_0	A_0	0	0
A_1	X_3	—	A_1	2	0
A_1	X_4	B_2	A_2	3	0
A_1	X_5	B_3	A_3	4	2-sb$_0$
A_1	X_6	B_4	A_4	4	2
A_2	$X_1 \cup X_2$	B_0	A_0	0	0
A_2	$X_3 \cup X_4$	—	A_2	3	0
A_2	X_5	B_3	A_3	4	2-sb$_0$
A_2	X_6	B_4	A_4	4	2
A_3	$X_1 \cup X_2$	B_0	A_0	0	0
A_3	$X_3 \cup X_4 \cup X_5$	—	A_3	4	2-sb$_0$
A_3	X_6	B_4	A_4	4	2
A_4	X_2	B_0	A_0	0	0
A_4	$\bigcup_{i\in\{1,3,4,5,6\}} X_i$	—	A_4	4	2

Table 9. Input Alphabet $\mathcal{A}_I$.

$\vec{c}_i$	V_{est}	V_{MRSP}	allowRevokeEB	X_i
$\vec{c}_1$	60	90	0	X_1
$\vec{c}_2$	60	90	1	X_2
$\vec{c}_3$	152	150	0	X_3
$\vec{c}_4$	125	120	1	X_4
$\vec{c}_5$	66	60	0	X_5
$\vec{c}_6$	260	230	0	X_6

7 Complete Test Suites for the CSM

7.1 Test Suite Construction – Overview

Applying the recipe for complete test strategies specified in Section 5.1.4, such a test suite can be constructed for the CSM as follows.

1. The input alphabet $\mathcal{A}_I$ consists of the vectors $\vec{c}_i$ selected from each of the IEC $\{X_1, \ldots, X_6\}$, as specified in Table 9.
2. For each of the two configurations "with service brake ($\mathrm{sb}_0 = 1$)", and "without service brake ($\mathrm{sb}_0 = 0$)" a DFSM is constructed with state space $\{q_0 = A_0 \cup A_1, q_2 = A_2, q_3 = A_3, q_4 = A_4\}$ and initial state $A_0 \cup A_1$; these I/O-equivalence classes have been identified above in Section 5.2.1. Each state machine operates on input alphabet $\mathcal{A}_I$ and output alphabet D_O, and their transition functions are specified by Table 8.

   The resulting DFSMs are shown in Figure 7, with $\mathrm{sb}_0 \in \{0, 1\}$ to be fixed according to the train configuration. In Table 10 the mapping of DFSM state names to I/O-equivalence classes is shown.
3. For each of the two DFSM a test suite is constructed according to the W-method, as described below.

Table 10. DFSM states and associated I/O-equivalence classes.

DFSM State	DFSM State Name	I/O-Equivalence Class
q_0	Normal or Overspeed	$A_0 \cup A_1$
q_2	Warning	A_2
q_3	Service Brake Intervention	A_3
q_4	Emergency Brake Intervention	A_4

7.2 Application of the W-Method

For application of the W-Method on the given DFSMs, we need to identify two sets of input traces, the *state-transition cover* STC and the *characterisation set* W [4, 29].

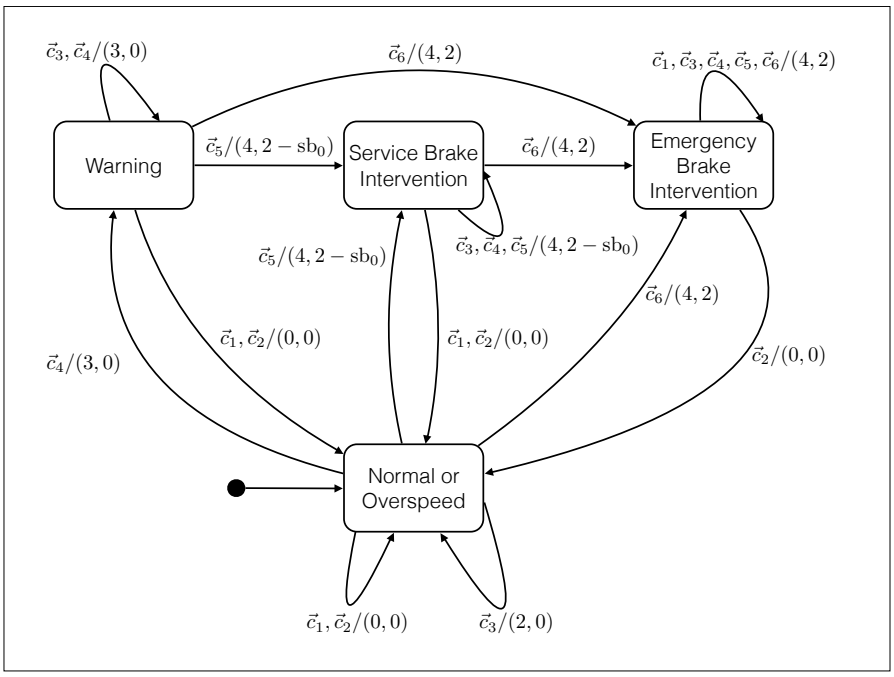
State-transition cover.

STC is constructed as a set of traces fulfilling

1. The empty trace is a member of STC.
2. For any reachable state q and any $\vec{c} \in \mathcal{A}_I$, there is an input trace $\iota \in STC$ such that
   - q can be reached from the initial state of the DFSM by application of ι, and
   - $\iota.\vec{c} \in STC$.

For the DFSMs under consideration, a state-transition cover is given by

$$STC = \{\varepsilon, \vec{c}_i, \vec{c}_j.\vec{c}_i \mid i = 1, \ldots, 6,\ j = 4, \ldots, 6\}$$



Output assignment actions (DMICmd, TICmd) = (α, β) are written as (α, β).

Figure 7. DFSM abstractions of the CSM, with configuration cases $\mathbf{sb}_0 \in \{0, 1\}$.

Characterisation set.

W is defined as a set of input traces distinguishing all DFSM states (recall that the DFSM under consideration are minimal) in the sense that for every pair of DFSM states q, q', there exists an input trace $\tau \in W$ such that τ applied to q yields an output sequence which differs from the one resulting from application of τ in state q'. For the case where the service brake differs from the emergency brake ($\mathrm{sb}_0 = 1$), the single input

$$W_{\mathrm{sb}_0=1} = \{\vec{c}_3\}$$

distinguishes all states of DFSM, as can be directly seen in Figure 7. In the case where the emergency brake is used for service brake intervention ($\mathrm{sb}_0 = 0$), we need additional input $\vec{c}_1$ to distinguish DFSM states q_3 and q_4.

$$W_{\mathrm{sb}_0=0} = \{\vec{c}_1, \vec{c}_3\}$$

Test suite according to W-Method.

With STC and W at hand, the W-Method asserts that the following test suite is complete for the fault domain of all DFSMs over the same input and output alphabets, whose state space has cardinality less or equal to m.

$$\mathcal{W}(STS) = STC.(\mathcal{A}_I)^{\max\{m,n\}-n}.W$$

We use notation $(\mathcal{A})^k$ to denote the subset of $\mathcal{A}^*$ containing all traces of length zero to k. $\mathcal{W}(STS)$ consists of all input traces starting with a (potentially empty) trace from the state-transition cover, continuing with a sequence of arbitrary inputs from the alphabet with length less or equal to $(\max\{m, n\} - n)$ (including the empty sequence), and ending with an input sequence contained in

the transition cover. $(\mathcal{A})^0$ just contains the empty trace, so $\mathcal{W}(STS)$ is reduced to $STC.W$, if the fault domain of DFSM whose state space is less or equal to that of the reference DFSM is considered.

Assuming that the minimal DFSM associated with the SUT implements at most two additional states ($m - n \leq 2$ implies $m \leq 6$), the W-Method produces the following test suite for the the minimal DFSM:

$$
\begin{aligned}
STC.(\mathcal{A}_I)^2.W_{\mathrm{sb}_0=1} \;=\; & \{\vec{c}_3\} \cup \\
& \{\vec{c}_i.\vec{c}_3 \mid i = 1,\ldots,6\} \cup \\
& \{\vec{c}_i.\vec{c}_j.\vec{c}_3 \mid i,j = 1,\ldots,6\} \cup \\
& \{\vec{c}_i.\vec{c}_j.\vec{c}_k.\vec{c}_3 \mid i,j,k = 1,\ldots,6\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_3 \mid i = 1,\ldots,6,\;\; j = 4,\ldots,6\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_3 \mid i,k = 1,\ldots,6,\;\; j = 4,\ldots,6\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_3 \mid h,i,k = 1,\ldots,6,\;\; j = 4,\ldots,6\}
\end{aligned}
$$

$$
\begin{aligned}
STC.(\mathcal{A}_I)^2.W_{\mathrm{sb}_0=0} \;=\; & \{\vec{c}_g \mid g = 1,3\} \cup \\
& \{\vec{c}_i.\vec{c}_g \mid i = 1,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_i.\vec{c}_j.\vec{c}_g \mid i,j = 1,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_i.\vec{c}_j.\vec{c}_h.\vec{c}_g \mid h,i,j = 1,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_g \mid i = 1,\ldots,6,\;\; j = 4,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_g \mid i,k = 1,\ldots,6,\;\; j = 4,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_g \mid h,i,k = 1,\ldots,6,\;\; j = 4,\ldots,6,\; g = 1,3\}
\end{aligned}
$$

Since ι-equivalence implies τ-equivalence when τ is a prefix of ι, the test suite produced by the W-Method can be reduced to the following:

$$
\begin{aligned}
\mathrm{TEST\_SUITE}_{\mathrm{sb}_0=1} \;=\; & \{\vec{c}_i.\vec{c}_j.\vec{c}_k.\vec{c}_3 \mid i,j,k = 1,\ldots,6\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_3 \mid h,i,k = 1,\ldots,6,\;\; j = 4,\ldots,6\}
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{TEST\_SUITE}_{\mathrm{sb}_0=0} \;=\; & \{\vec{c}_i.\vec{c}_j.\vec{c}_h.\vec{c}_g \mid h,i,j = 1,\ldots,6,\; g = 1,3\} \cup \\
& \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_g \mid h,i,k = 1,\ldots,6,\;\; j = 4,\ldots,6,\; g = 1,3\}
\end{aligned}
$$

8 Test Strength

8.1 Test Strength Assessment

The notion of *test strength* refers to the capability of test suites to uncover errors in an SUT. The IECP testing strategy considered here is associated with a well-defined test strength, specified by means of the fault model: all deviations of an SUT from I/O-equivalence to the reference model $\mathcal{S}$ will be detected, provided that the true behaviour of the SUT can be specified by means of an STS which is contained in the fault domain. Based on the fault domain, the discussion of test strength is "transformed" into the question whether the size of the fault domain for which the test suite has been created is sufficient to contain all erroneous behaviours we can reasonably expect.

In the next three sections examples will be presented where the erroneous SUT behaviour can already by uncovered with IECP test suites based on the fault domain $\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$ specified in Section 6. The first example (Section 8.2) addresses a type of erroneous behaviour that could not only be uncovered by the equivalence class testing strategy described in this report, but is also guaranteed to be detected by test suites achieving transition coverage of the SysML state machine model (see Section 10). The second example (Section 8.3) shows a type of failure that is *not guaranteed* to be uncovered by test cases from a simple model coverage strategy like transition or MC/DC coverage, because it introduces an additional basic state in the SUT's SysML state machine. Practical test execution documented in Section 10, however, shows that the failure is "accidentally" uncovered with the transition coverage strategy by the model-based testing tool used, due to suitable input values selected by the tool when trying to cover the required transitions. The third example (Section 8.4) exhibits an even harder failure type, which cannot be detected by any of the transition-based coverage strategies, as is documented by the test results shown in Section 10.

In Section 9 we will discuss fault models associated with larger fault domains, based on true refinements of $\mathcal{I}$.

8.2 Example 1

Suppose that the implementation acts like the SysML state machine depicted in Fig. 8: from basic state OVERSPEED there is a transition failure in the SysML state machine which links to basic state EMER_BRAKE instead of NORMAL, when guard condition $[V_{est} \leq V_{MRSP}]$ evaluates to `true`. Note that the DFSM associated with STS' now has one more state than the model's DFSMs, because OVERSPEED and NORMAL are no longer equivalent.

Assuming the case $\mathrm{sb}_0 = 1$ and applying $\mathrm{TEST\_SUITE}_{\mathrm{sb}_0=1}$ introduced in Section 7, this failure will be uncovered, for example, by test case

$$\begin{aligned} \iota &= \vec{c}_1.\vec{c}_3.\vec{c}_1.\vec{c}_3 \\ &= (V_{est} = 60, V_{MRSP} = 90, \mathrm{allowRevokeEB} = 0).(152, 150, 0).(60, 90, 0).(152, 150, 0) \end{aligned}$$

On the model STS, this input will trigger output sequence

$$(\mathrm{DMICmd} = 0, \mathrm{TICmd} = 0).(2, 0).(0, 0).(2, 0)$$

whereas output sequence

$$(\mathrm{DMICmd} = 0, \mathrm{TICmd} = 0).(2, 0).(4, 2).(4, 2)$$

will be triggered by ι, when applied to STS' representing the SUT.

8.3 Example 2

Suppose that the implementation has an additional control state NORMAL_2, as depicted in Fig. 9. When the SUT is in the control state OVERSPEED and inputs satisfy $V_{est} \leq V_{MRSP}$, the SUT's state machine transits to the new basic state NORMAL_2. For this control state, there is only one outgoing transition whose guard condition is $V_{est} > V_{MRSP} + dV_{\mathrm{ebi}}$, and target control state is EMER_BRAKE. As a consequence, the SUT fails to issue warnings and to trigger service brake intervention, after having entered NORMAL_2; only the emergency brake intervention condition is handled correctly. The SUT failure remains hidden until a transition sequence from NORMAL to OVERSPEED and back to NORMAL should be performed. Note that the introduction of this additional basic state in the SUT's SysML state machine leads to a DFSM abstraction that has *two* more states than the DFSM abstraction of the original model,

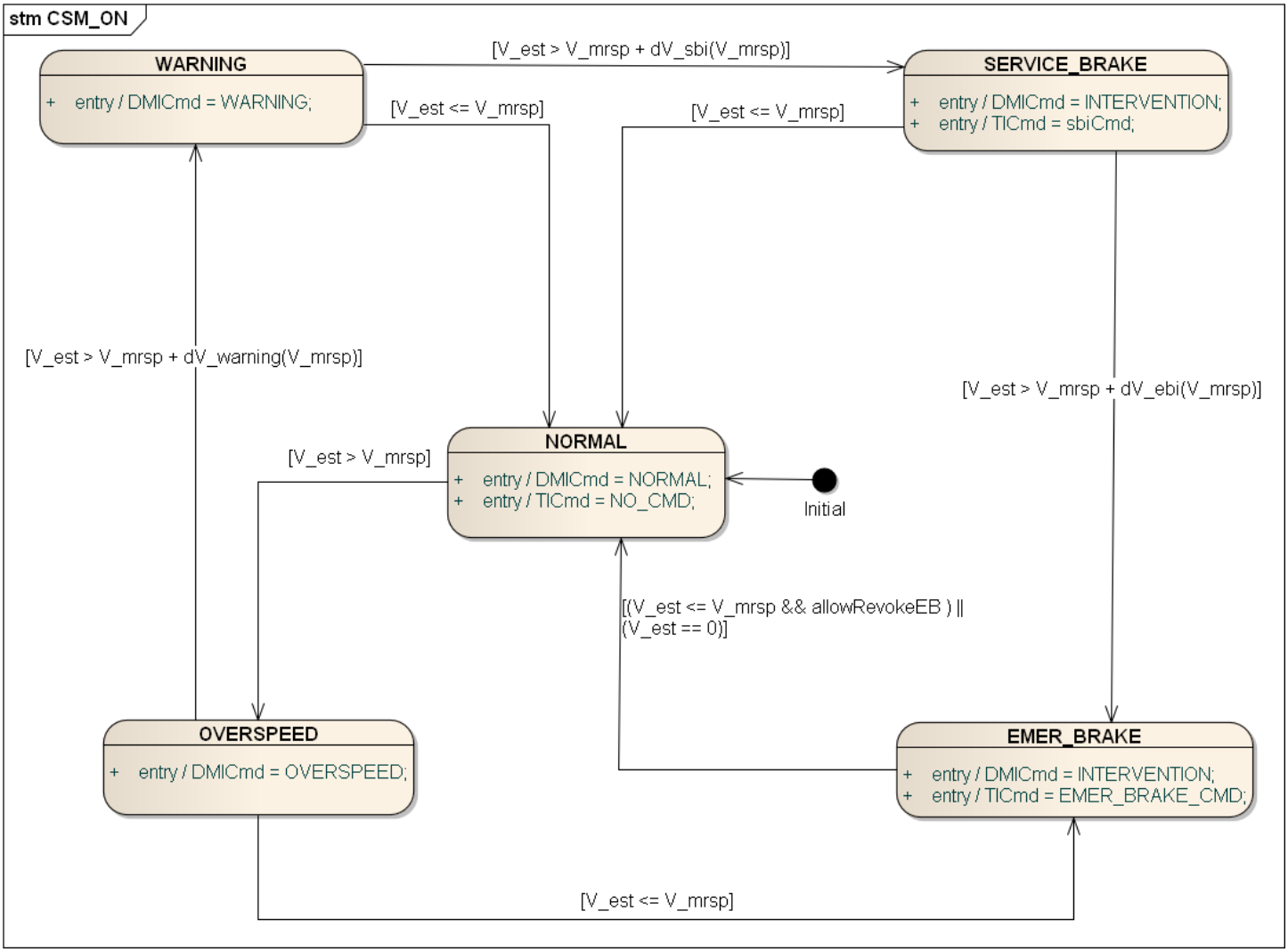


Figure 8. Faulty SUT – Example 1.

because basic states NORMAL and OVERSPEED are no longer equivalent, and NORMAL_2 is non-equivalent to any of the other states.

Assuming again the case $sb_0 = 1$ and applying $\text{TEST\_SUITE}_{sb_0=1}$ introduced in Section 7, this failure will be uncovered, for example, by test case

$$\begin{aligned} \iota &= \vec{c}_3.\vec{c}_1.\vec{c}_4.\vec{c}_3 \\ &= (V_{est} = 152, V_{MRSP} = 150, \text{allowRevokeEB} = 0).(60, 90, 0).(125, 120, 1).(152, 150, 0) \end{aligned}$$

On the model STS, this input will trigger output sequence

$$(\text{DMICmd} = 2, \text{TICmd} = 0).(0, 0).(3, 0).(3, 0)$$

whereas output sequence

$$(\text{DMICmd} = 2, \text{TICmd} = 0).(0, 0).(0, 0).(0, 0)$$

will be triggered by ι, when applied to STS' representing the SUT.

Observe that this SUT failure would not be uncovered by every ordinary transition coverage test strategy based on SysML state machine model. Transition coverage would be achieved, for example, by the four test cases

$$\text{TC\_TEST\_SUITE} = \{\vec{c}_3.\vec{c}_1, \vec{c}_3.\vec{c}_4.\vec{c}_1, \vec{c}_4.\vec{c}_5.\vec{c}_1, \vec{c}_5.\vec{c}_6.\vec{c}_2\}$$

which fail to uncover the violation of I/O-equivalence.

8.4 Example 3

Suppose that the implementation has an additional control state NORMAL_2 as shown in the mutant in Figure 10. When the SUT is in the control state Warning and inputs satisfy $V_{est} \leq V_{MRSP}$, the SUT's state machine transits to the new basic state NORMAL_2. For this control state, there is only one outgoing transition in SysML whose guard condition is $V_{est} > V_{MRSP} + dV_{\text{warning}}$, and target control state is Warning. Note that the introduction of this additional basic state in the SUT's SysML state machine leads to a DFSM abstraction that has *two* more states than the DFSM abstraction of the original model, because basic states NORMAL and OVERSPEED are no longer equivalent, and NORMAL_2 is non-equivalent to any of the other states.

Assuming again the case $sb_0 = 1$ and applying $\text{TEST\_SUITE}_{sb_0=1}$ introduced in Section 7, this failure will be uncovered, for example, by test case

$$\begin{aligned} \iota &= \vec{c}_4.\vec{c}_1.\vec{c}_3 \\ &= (V_{est} = 125, V_{MRSP} = 120, \text{allowRevokeEB} = 1).(60, 90, 0).(152, 150, 0) \end{aligned}$$

On the model STS, this input will trigger output sequence

$$(\text{DMICmd} = 3, \text{TICmd} = 0).(0, 0).(2, 0)$$

whereas output sequence

$$(\text{DMICmd} = 3, \text{TICmd} = 0).(0, 0).(0, 0)$$

will be triggered by ι, when applied to STS' representing the SUT.

Observe that this SUT failure would not be uncovered by every ordinary transition coverage test strategy based on SysML state machine model. Transition coverage would be achieved, for example, by the test procedure TP-002 in Section 10 which fail to uncover the violation of I/O-equivalence.

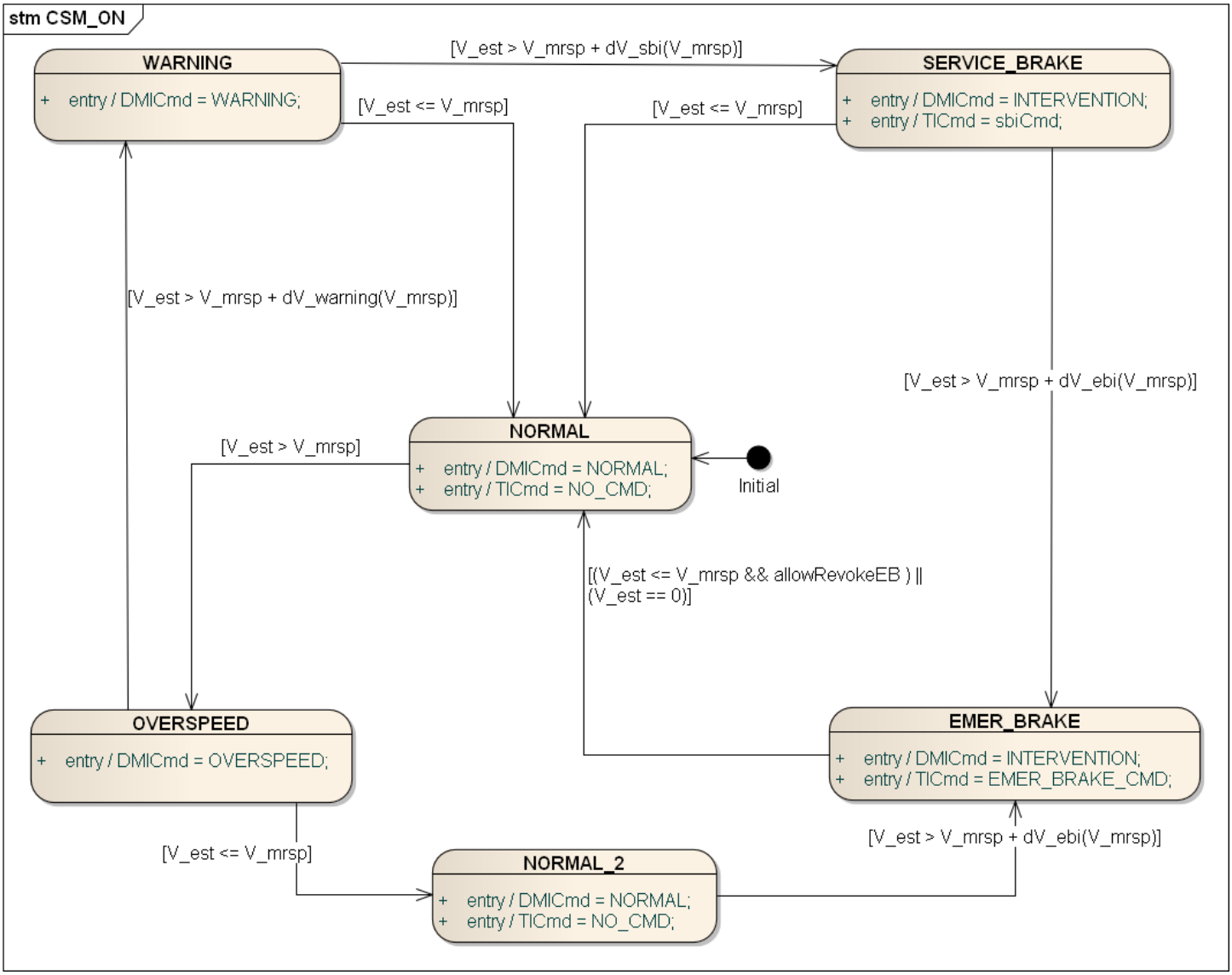


Figure 9. Faulty SUT – Example 2.

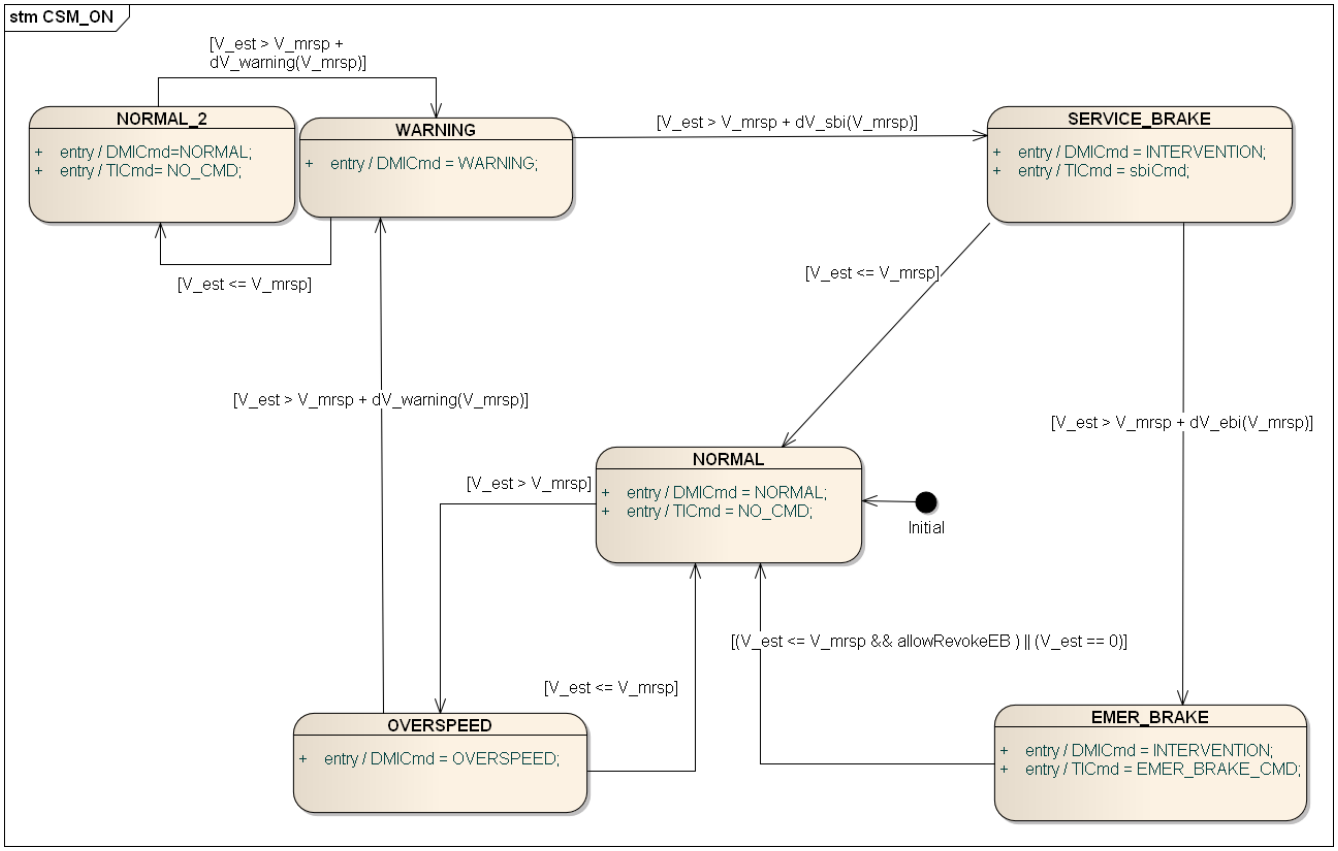


Figure 10. Faulty SUT – Example 3.

9 Heuristics for Constructing IECP Refinements

9.1 IECP Refinements for the CSM

The definition of IECP refinements given in Section 5.1.2 induces a construction mechanism based on propositions. This is applied to the CSM as follows.

1. Choose a new proposition γ over free variables from I.
2. Strengthen each proposition $\Phi_k,\ k = 1, \ldots, 6$ defining an IEC by

$$\begin{aligned} \Phi^I_{k,+} &\equiv \Phi_k \wedge \gamma \\ \Phi^I_{k,-} &\equiv \Phi_k \wedge \neg\gamma \end{aligned}$$

3. Specify the refinement by

$$\begin{aligned} \mathcal{I}_2 &= \{X_{k,+}, X_{k,-} \mid k = 1, \ldots, 6\} \\ X_{k,+} &= \{\vec{c} \in D_I \mid \Phi_{k,+}[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]\} \\ X_{k,-} &= \{\vec{c} \in D_I \mid \Phi_{k,-}[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]\} \end{aligned}$$

4. Refine the old input alphabet by adding input vectors for each new $X_{k,+}, X_{k,-}$ which does not have a representative in the old alphabet.

Refinements of the original IECP are needed, for example, when it is suspected that the SUT implements so-called *trapdoors* [1]: these are transitions whose trigger conditions are refinements

$\varphi^I_{q,i} \wedge \gamma$ of trigger conditions $\varphi^I_{q,i}$ in the original model. The SUT behaviour conforms to the associated model transition for inputs satisfying $\varphi^I_{q,i} \wedge \neg\gamma$, but shows erroneous behaviour for inputs satisfying $\varphi^I_{q,i} \wedge \gamma$. Trapdoors may occur in transitions from quiescent to quiescent, and in transitions from quiescent to transient states. In the subsequent sections this and other motivations for introducing refinements are discussed.

9.2 Overview of the Refinement Concept

In this section we present and discuss a heuristic for constructing IECP refinements resulting in larger fault domains that are well-justified from the perspective of standards related to safety-critical systems development in the avionic, automotive, and railway domains [26, 30, 14, 3].

Our heuristic involves the following refinement steps, and we suggest to apply these in the order presented here.

1. Requirements-based IECP refinement
2. Boundary value IECP refinement
3. IECP refinement by sub-paving

9.3 Requirements-based IECP Refinement

9.3.1 Requirements-related Case Distinctions

In Section 4.7.5 we introduced the transient state input conditions $\varphi^I_{q,i}$. Intuitively speaking, each of these conditions, when feasible and applied in quiescent state class q, triggers a well-defined behaviour, namely the transformation of internal state and outputs performed by the transition from transient state class B_i to its quiescent successor state class A_i. Since this transformation is constant (it only depends on B_i), it reflects one system requirement or a part of it. As a consequence, any disjunction occurring in such a condition $\varphi^I_{q,i}$ reflects a case distinction for the same requirement.

Example Condition

$$\varphi^I_{0,1}(V) \equiv V_{MRSP} < V_{est} \wedge V_{est} \leq V_{MRSP} + \mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$$

reflects the pre-condition for requirement REQ-3.13.10.3.3.t2 (see Table 2), concerning the transition from NORMAL state to OVERSPEED state in the SysML state machine, with activation of the overspeed indication at the driver machine interface. $\varphi^I_{0,1}(V)$ is applied in A_0, and triggers a transition into B_1 from where the overspeed indication is activated and the system stabilises again in A_1. Expanding $\mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$, we get

$$\begin{aligned}\varphi^I_{0,1}(V) \quad \equiv \quad & V_{MRSP} < V_{est} \wedge \\ & ((V_{MRSP} \leq 110 \wedge V_{est} \leq V_{MRSP} + 4) \vee \\ & (110 < V_{MRSP} \leq 140 \wedge V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3}) \vee \\ & (140 < V_{MRSP} \wedge V_{est} \leq V_{MRSP} + 5))\end{aligned}$$

The disjunction inside the second conjunct covers the three case distinctions for transiting into OVERSPEED (see definition of $\mathrm{dV}_{\mathsf{warning}}(V_{MRSP})$ in Equation (2)). □

Similarly, the quiescent state conditions $\varphi_i^I(I) = \varphi_{i,i}^I(I)$ specify stability requirements regarding input changes that will not lead to any new system reaction, so disjunctions in $\varphi_{i,i}^I(I)$ represent case distinctions of these stability requirements.

Example Condition

$$\varphi_4^I(V) \equiv \varphi_{4,4}^I(V) \equiv (0 < V_{est} \wedge \text{allowRevokeEB} = 0) \vee (V_{MRSP} < V_{est} \wedge \text{allowRevokeEB} = 1)$$

describes the stability condition when in SysML state machine state EMER_BRAKE. There are two cases to consider.

1. When the emergency brake command may only be revoked after the train has come to a standstill (case allowRevokeEB = 0), any input condition satisfying $0 < V_{est}$ is a stability condition when in basic state EMER_BRAKE.
2. When the emergency brake command may already be revoked as soon as $V_{est} \leq V_{MRSP}$ (case allowRevokeEB = 1), any input condition satisfying $V_{MRSP} < V_{est}$ is a stability condition when in basic state EMER_BRAKE.

□

Standards for safety-relevant systems always demand that requirements should be completely covered by tests (see, e.g.,[30, 6.4.4]). Therefore it is advisable to refine the IECP in such a way, that the extended input alphabet resulting from this refinement covers all requirements-related case distinctions.

9.3.2 Construction of the Requirements-based IECP Refinement

The mechanisable construction of the requirements-based IECP refinement is performed for the CSM as follows.

1. Transform each $\varphi_{q,i}^I$ and each $\varphi_{i,i}^I$ into disjunctive normal form (DNF). This results in disjuncts $\varphi_{q,i,h}^I$, such that
$$\varphi_{q,i}^I \equiv \bigvee_j \varphi_{q,i,h}^I$$
Define index set I_q as the set of all pairs (i, h) where $\varphi_{q,i,h}^I$ is a disjunct in the DNF of $\varphi_{q,i}^I$.
2. Drop infeasible disjuncts. This can be performed by means of an SMT solver determining for each disjunct whether it has at least one solution.
3. Refine the $\Phi_k,\ k = 1, \ldots, 6$ specifying the IECP of the CSM as described in Section 5.2.2 to propositions
$$\Phi_{k,(j_0,h_0),\ldots,(j_4,h_4)} \equiv \bigwedge_{q=0}^{4} \varphi_{q,j_q,h_q}^I$$
where $(j_q, h_q) \in I_q,\ q = 0, \ldots, 4$.
4. Create IECs in analogy to the rules given in Section 5.1.2, but this time using feasible solutions $\Phi_{k,(j_0,h_0),\ldots,(j_4,h_4)}$:
$$X_{k,(j_0,h_0),\ldots,(j_4,h_4)} = \{\vec{c} \in D_I \mid \Phi_{k,(j_0,h_0),\ldots,(j_4,h_4)}[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]\}$$

5. Determine the $X_{k,(j_0,h_0),\ldots,(j_4,h_4)}$ that are already represented by the existing members $\vec{c}$ of the old alphabet.

6. For each of the remaining $X_{k,(j_0,h_0),\ldots,(j_4,h_4)}$, select a new member for the new, extended input alphabet by letting a constraint solver find a solution $\vec{c}$ for

$$\Phi_{k,(j_0,h_0),\ldots,(j_4,h_4)}[\vec{c}/(V_{est}, V_{MRSP}, \text{allowRevokeEB})]$$

Example For the CSM, the requirements-based IECP-refinement leads to the following disjuncts $\varphi^I_{q,i,j}$.

$$\begin{aligned}
\varphi^I_{0,0,0} &\equiv \varphi^I_0(V) \equiv V_{est} \leq V_{MRSP} \\
\varphi^I_{1,1,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 4 \\
\varphi^I_{1,1,1} &\equiv 110 < V_{MRSP} \leq 140 \wedge V_{MRSP} < V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\varphi^I_{1,1,2} &\equiv 140 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5 \\
\varphi^I_{2,2,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5.5 \\
\varphi^I_{2,2,1} &\equiv 110 < V_{MRSP} \leq 210 \wedge V_{MRSP} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\varphi^I_{2,2,2} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 10 \\
\varphi^I_{3,3,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 7.5 \\
\varphi^I_{3,3,1} &\equiv 110 < V_{MRSP} \leq 210 \wedge V_{MRSP} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\varphi^I_{3,3,2} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 15 \\
\varphi^I_{4,4,0} &\equiv 0 < V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 0 \\
\varphi^I_{4,4,1} &\equiv V_{MRSP} < V_{est} \\
\varphi^I_{0,1,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 4 \\
\varphi^I_{0,1,1} &\equiv 110 < V_{MRSP} \leq 140 \wedge V_{MRSP} < V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\varphi^I_{0,1,2} &\equiv 140 < V_{MRSP} \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5 \\
\varphi^I_{0,2,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 4 < V_{est} \leq V_{MRSP} + 5.5 \\
\varphi^I_{0,2,1} &\equiv 110 < V_{MRSP} \leq 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\varphi^I_{0,2,2} &\equiv 140 < V_{MRSP} \leq 210 \wedge V_{MRSP} + 5 < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\varphi^I_{0,2,3} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 5 < V_{est} \leq V_{MRSP} + 10 \\
\varphi^I_{0,3,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 5.5 < V_{est} \leq V_{MRSP} + 7.5 \\
\varphi^I_{0,3,1} &\equiv 110 < V_{MRSP} \leq 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\varphi^I_{0,3,2} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 10 < V_{est} \leq V_{MRSP} + 15 \\
\varphi^I_{0,4,0} &\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 7.5 < V_{est} \\
\varphi^I_{0,4,1} &\equiv 110 < V_{MRSP} \leq 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est} \\
\varphi^I_{0,4,2} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 15 < V_{est}
\end{aligned}$$

$$
\begin{aligned}
\varphi^I_{1,2,i} &\equiv \varphi^I_{0,2,i},\ i = 0,1,2,3 \\
\varphi^I_{1,3,i} &\equiv \varphi^I_{0,3,i},\ i = 0,1,2 \\
\varphi^I_{1,4,i} &\equiv \varphi^I_{0,4,i},\ i = 0,1,2 \\
\varphi^I_{2,3,i} &\equiv \varphi^I_{0,3,i}\ i = 0,1,2 \\
\varphi^I_{2,4,i} &\equiv \varphi^I_{0,4,i}\ i = 0,1,2 \\
\varphi^I_{3,4,i} &\equiv \varphi^I_{0,4,i}\ i = 0,1,2 \\
\varphi^I_{1,0,0} &\equiv \varphi_{1,0} \equiv V_{est} \leq V_{MRSP} \\
\varphi^I_{2,0,0} &\equiv \varphi^I_{1,0,0} \\
\varphi^I_{3,0,0} &\equiv \varphi^I_{1,0,0} \\
\varphi^I_{4,0,0} &\equiv V_{est} = 0 \\
\varphi^I_{4,0,1} &\equiv 0 < V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 1
\end{aligned}
$$

This leads to the following refined predicates Φ_{i_j}.

$$
\begin{aligned}
\Phi_{1_0} &\equiv \varphi^I_{0,0,0} \wedge \varphi^I_{1,0,0} \wedge \varphi^I_{2,0,0} \wedge \varphi^I_{3,0,0} \wedge \varphi^I_{4,4,0} \\
&\equiv \varphi^I_{0,0,0} \wedge \varphi^I_{4,4,0} \\
&\equiv 0 < V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 0 \\
\Phi_{2_0} &\equiv \varphi^I_{0,0,0} \wedge \varphi^I_{1,0,0} \wedge \varphi^I_{2,0,0} \wedge \varphi^I_{3,0,0} \wedge \varphi^I_{4,0,0} \\
&\equiv \varphi^I_{4,0,0} \\
&\equiv V_{est} = 0 \\
\Phi_{2_1} &\equiv \varphi^I_{0,0,0} \wedge \varphi^I_{1,0,0} \wedge \varphi^I_{2,0,0} \wedge \varphi^I_{3,0,0} \wedge \varphi^I_{4,0,1} \\
&\equiv \varphi^I_{4,0,1} \\
&\equiv 0 < V_{est} \leq V_{MRSP} \wedge \text{allowRevokeEB} = 1 \\
\Phi_{3_0} &\equiv \varphi^I_{0,1,0} \wedge \varphi^I_{1,1,0} \wedge \varphi^I_{2,2,0} \wedge \varphi^I_{3,3,0} \wedge \varphi^I_{4,4,1} \\
&\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 4 \\
\Phi_{3_1} &\equiv \varphi^I_{0,1,1} \wedge \varphi^I_{1,1,1} \wedge \varphi^I_{2,2,1} \wedge \varphi^I_{3,3,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 110 < V_{MRSP} \leq 140 \wedge V_{MRSP} < V_{est} \leq \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\Phi_{3_2} &\equiv \varphi^I_{0,1,2} \wedge \varphi^I_{1,1,2} \wedge \varphi^I_{2,2,1} \wedge \varphi^I_{3,3,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 140 < V_{MRSP} \leq 210 \wedge V_{MRSP} < V_{est} \leq V_{MRSP} + 5
\end{aligned}
$$

$$
\begin{aligned}
\Phi_{3_3} &\equiv \varphi^I_{0,1,2} \wedge \varphi^I_{1,1,2} \wedge \varphi^I_{2,2,2} \wedge \varphi^I_{3,3,2} \wedge \varphi^I_{4,4,1} \\
&\equiv 210 < V_{MRSP} < V_{est} \leq V_{MRSP} + 5 \\
\Phi_{4_0} &\equiv \varphi^I_{0,2,0} \wedge \varphi^I_{1,2,0} \wedge \varphi^I_{2,2,0} \wedge \varphi^I_{3,3,0} \wedge \varphi^I_{4,4,1} \\
&\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 4 < V_{est} \leq V_{MRSP} + 5.5 \\
\Phi_{4_1} &\equiv \varphi^I_{0,2,1} \wedge \varphi^I_{1,2,1} \wedge \varphi^I_{2,2,1} \wedge \varphi^I_{3,3,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 110 < V_{MRSP} \leq 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_2} &\equiv \varphi^I_{0,2,2} \wedge \varphi^I_{1,2,2} \wedge \varphi^I_{2,2,1} \wedge \varphi^I_{3,3,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 140 < V_{MRSP} \leq 210 \wedge V_{MRSP} + 5 < V_{est} \leq \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_3} &\equiv \varphi^I_{0,2,3} \wedge \varphi^I_{1,2,3} \wedge \varphi^I_{2,2,2} \wedge \varphi^I_{3,3,2} \wedge \varphi^I_{4,4,1} \\
&\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 5 < V_{est} \leq V_{MRSP} + 10 \\
\Phi_{5_0} &\equiv \varphi^I_{0,3,0} \wedge \varphi^I_{1,3,0} \wedge \varphi^I_{2,3,0} \wedge \varphi^I_{3,3,0} \wedge \varphi^I_{4,4,1} \\
&\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 5.5 < V_{est} \leq V_{MRSP} + 7.5 \\
\Phi_{5_1} &\equiv \varphi^I_{0,3,1} \wedge \varphi^I_{1,3,1} \wedge \varphi^I_{2,3,1} \wedge \varphi^I_{3,3,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 110 < V_{MRSP} \leq 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} \leq \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\Phi_{5_2} &\equiv \varphi^I_{0,3,2} \wedge \varphi^I_{1,3,2} \wedge \varphi^I_{2,3,2} \wedge \varphi^I_{3,3,2} \wedge \varphi^I_{4,4,1} \\
&\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 10 < V_{est} \leq V_{MRSP} + 15 \\
\Phi_{6_0} &\equiv \varphi^I_{0,4,0} \wedge \varphi^I_{1,4,0} \wedge \varphi^I_{2,4,0} \wedge \varphi^I_{3,4,0} \wedge \varphi^I_{4,4,1} \\
&\equiv V_{MRSP} \leq 110 \wedge V_{MRSP} + 7.5 < V_{est} \\
\Phi_{6_1} &\equiv \varphi^I_{0,4,1} \wedge \varphi^I_{1,4,1} \wedge \varphi^I_{2,4,1} \wedge \varphi^I_{3,4,1} \wedge \varphi^I_{4,4,1} \\
&\equiv 110 < V_{MRSP} \leq 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est} \\
\Phi_{6_2} &\equiv \varphi^I_{0,4,2} \wedge \varphi^I_{1,4,2} \wedge \varphi^I_{2,4,2} \wedge \varphi^I_{3,4,2} \wedge \varphi^I_{4,4,1} \\
&\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 15 < V_{est}
\end{aligned}
$$

From the refined predicates additional members of the input alphabet are selected. This results in a new alphabet as the one shown in Table 11.

The resulting test suite comprises the test cases of the old suite plus additional cases, the new test suite can be expressed as an extension of the old one. Since $(A \cup B)^3 = A^3 \cup A^2.B \cup A.B.(A \cup B) \cup B.(A \cup B)^2$, for any subsets A, B of the input alphabet, the old input alphabet with set $A = \{c_1, \ldots, c_6\}$, and the new alphabet values contained in $B = \{c_7, \ldots, c_{17}\}$ result in the following test suite.

$$
\begin{aligned}
\text{TEST\_SUITE}_{sb_0=1}{}' &= \{\vec{c}_i.\vec{c}_j.\vec{c}_k.\vec{c}_3 \mid i, j, k = 1, \ldots, 17\} \cup \\
&\quad \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_3 \mid h, i, k = 1, \ldots, 17, j = 4, 5, 6\} \cup \\
&= \{\vec{c}_i.\vec{c}_j.\vec{c}_k.\vec{c}_3 \mid i, j, k = 1, \ldots, 6\} \cup \\
&\quad \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_3 \mid h, i, k = 1, \ldots, 6, j = 4, 5, 6\} \cup \\
&\quad \{\vec{c}_i.\vec{c}_j.\vec{c}_k.\vec{c}_3 \mid i, j = 1, \ldots, 6, k = 7, \ldots, 17\} \cup \\
&\quad \{\vec{c}_i.\vec{c}_k.\vec{c}_j.\vec{c}_3 \mid i = 1, \ldots, 6, j = 1, \ldots, 17, k = 7, \ldots, 17\} \cup \\
&\quad \{\vec{c}_k.\vec{c}_i.\vec{c}_j.\vec{c}_3 \mid i, j = 1, \ldots, 17, k = 7, \ldots, 17\} \cup \\
&\quad \{\vec{c}_j.\vec{c}_i.\vec{c}_h.\vec{c}_k.\vec{c}_3 \mid i, h = 1, \ldots, 6, k = 7, \ldots, 17, j = 4, 5, 6\} \cup \\
&\quad \{\vec{c}_j.\vec{c}_i.\vec{c}_k.\vec{c}_h.\vec{c}_3 \mid i = 1, \ldots, 6, h = 1, \ldots, 17, k = 7, \ldots, 17, j = 4, 5, 6\} \cup \\
&\quad \{\vec{c}_j.\vec{c}_k.\vec{c}_i.\vec{c}_h.\vec{c}_3 \mid i, h = 1, \ldots, 17, k = 7, \ldots, 17, j = 4, 5, 6\}
\end{aligned}
$$

□

Table 11. Extended input alphabet $\mathcal{A}'_I$ satisfying $\mathcal{A}_I \subseteq \mathcal{A}'_I$.

$\vec{c}_i$	V_{MRSP}	V_{est}	allowRevokeEB	X_i	X_{i_j}
$\vec{c}_1$	90	60	0	X_1	X_{1_0}
$\vec{c}_2$	90	60	1	X_2	X_{2_1}
$\vec{c}_3$	150	152	0	X_3	X_{3_2}
$\vec{c}_4$	120	125	1	X_4	X_{4_1}
$\vec{c}_5$	60	66	0	X_5	X_{5_0}
$\vec{c}_6$	230	260	1	X_6	X_{6_2}
$\vec{c}_7$	90	0	1	X_2	X_{2_0}
$\vec{c}_8$	90	93	0	X_3	X_{3_0}
$\vec{c}_9$	112	115	1	X_3	X_{3_1}
$\vec{c}_{10}$	211	212	0	X_3	X_{3_3}
$\vec{c}_{11}$	90	95	1	X_4	X_{4_0}
$\vec{c}_{12}$	150	156	0	X_4	X_{4_2}
$\vec{c}_{13}$	220	226	1	X_4	X_{4_3}
$\vec{c}_{14}$	205	215	0	X_5	X_{5_1}
$\vec{c}_{15}$	230	244	1	X_5	X_{5_2}
$\vec{c}_{16}$	55	100	0	X_6	X_{6_0}
$\vec{c}_{17}$	200	215	1	X_6	X_{6_1}

9.3.3 Discussion

The requirements-based IECP refinement ensures that all case distinctions of requirements are tested. As a consequence, all SUT failures will be detected, where in a certain SUT state all inputs that should trigger the reaction for a given requirements sub-case are handled incorrectly. Compared to the original IECP, this is a considerable improvement, because the original IECP is only guaranteed to detect failures if *all* sub-cases of a requirement are incorrectly handled by the SUT in a certain state.

Just as for the original IECP, however, the requirements-based IECP refinement relies on the SUT implementing all control decisions, that is, all guard conditions specified in the original SysML state machine, correctly. Failures are only allowed in the actions setting outputs and internal states. This is of considerable value if SUT code is generated semi-automatically. A qualified code-frame generator, for example, might derive a control procedure from the state machine model, so that all control states, guard conditions, and state machine transitions are generated with high reliability by the tool. However, the driver software implementing access to the hardware output interfaces – in our example the DMI and the train interface acting on the brakes – may have been implemented in a manual way, so that some actions will show erroneous behaviour in certain states of the SUT.

For other development scenarios, in particular, when the SUT code has been developed from the requirements in manual way, the test strength of the requirements-based IECP refinement is still insufficient, because SUT failures in the handling of guard conditions are just as likely as failures in the action implementations. These failure scenarios will be considered in the next refinements.

9.4 Boundary Value IECP Refinement

Let $\mathcal{I}_2 = \{X_{1_0}, X_{2_0}, X_{2_1}, \ldots, X_{6_2}\}$ be the input equivalence class partitioning containing the 17 input equivalence classes constructed above by Φ_{i_j}. Extend the old input alphabet to a new input alphabet which includes input vectors of the boundary of each X_{i_j}. Then the new input alphabet (see Table 12) contains 31 inputs more then the old one, and the new test suite contains $4 \cdot 48^3 = 442368$ test cases, $4 \cdot 48^3 - 4 \cdot 17^3 = 442368 - 19652 = 422716$ test cases more.

Since the CSM code does not depend on dealt-time conditions, this is still an acceptable number of software tests. For hardware-in-the-loop tests, when timing constraints of the target hardware and the operational environment have to be considered, however, it would be too time-consuming to execute them all. A well-justified reduction heuristic in this case is to drop the tests of certain boundary values for states where these boundaries are of no relevance.

The following refined predicates $\Phi_{i_j,k}$ contain the separate boundary conditions.

$$
\begin{aligned}
\Phi_{1_0,0} &\equiv 0 < V_{est} < V_{MRSP} \wedge \text{allowRevokeEB} = 0 \\
\Phi_{1_0,1} &\equiv 0 < V_{est} = V_{MRSP} \wedge \text{allowRevokeEB} = 0 \\
\Phi_{2_0} &\equiv V_{est} = 0 \\
\Phi_{2_1,0} &\equiv 0 < V_{est} < V_{MRSP} \wedge \text{allowRevokeEB} = 1 \\
\Phi_{2_1,1} &\equiv V_{est} = V_{MRSP} \wedge \text{allowRevokeEB} = 1 \\
\Phi_{3_0,0} &\equiv V_{MRSP} < 110 \wedge V_{MRSP} < V_{est} < V_{MRSP} + 4 \\
\Phi_{3_0,1} &\equiv V_{MRSP} = 110 \wedge V_{MRSP} < V_{est} < V_{MRSP} + 4 \\
\Phi_{3_0,2} &\equiv V_{MRSP} < 110 \wedge V_{est} = V_{MRSP} + 4 \\
\Phi_{3_0,3} &\equiv V_{MRSP} = 110 \wedge V_{est} = V_{MRSP} + 4 \\
\Phi_{3_1,0} &\equiv 110 < V_{MRSP} < 140 \wedge V_{MRSP} < V_{est} < \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\Phi_{3_1,1} &\equiv V_{MRSP} = 140 \wedge V_{MRSP} < V_{est} < \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\Phi_{3_1,2} &\equiv 110 < V_{MRSP} < 140 \wedge V_{est} = \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\Phi_{3_1,3} &\equiv V_{MRSP} = 140 \wedge V_{MRSP} < V_{est} = \frac{31}{30} V_{MRSP} + \frac{1}{3} \\
\Phi_{3_2,0} &\equiv 140 < V_{MRSP} < 210 \wedge V_{MRSP} < V_{est} < V_{MRSP} + 5 \\
\Phi_{3_2,1} &\equiv V_{MRSP} = 210 \wedge V_{MRSP} < V_{est} < V_{MRSP} + 5 \\
\Phi_{3_2,2} &\equiv 140 < V_{MRSP} < 210 \wedge V_{est} = V_{MRSP} + 5 \\
\Phi_{3_2,3} &\equiv V_{MRSP} = 210 \wedge V_{est} = V_{MRSP} + 5 \\
\Phi_{3_3,0} &\equiv 210 < V_{MRSP} < V_{est} < V_{MRSP} + 5 \\
\Phi_{3_3,1} &\equiv 210 < V_{MRSP} < V_{est} = V_{MRSP} + 5 \\
\Phi_{4_0,0} &\equiv V_{MRSP} < 110 \wedge V_{MRSP} + 4 < V_{est} < V_{MRSP} + 5.5 \\
\Phi_{4_0,1} &\equiv V_{MRSP} = 110 \wedge V_{MRSP} + 4 < V_{est} < V_{MRSP} + 5.5 \\
\Phi_{4_0,2} &\equiv V_{MRSP} < 110 \wedge V_{est} = V_{MRSP} + 5.5 \\
\Phi_{4_0,3} &\equiv V_{MRSP} = 110 \wedge V_{est} = V_{MRSP} + 5.5 \\
\Phi_{4_1,0} &\equiv 110 < V_{MRSP} < 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} < \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_1,1} &\equiv V_{MRSP} = 140 \wedge \frac{31}{30} V_{MRSP} + \frac{1}{3} < V_{est} < \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_1,2} &\equiv 110 < V_{MRSP} < 140 \wedge V_{est} = \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_1,3} &\equiv V_{MRSP} = 140 \wedge V_{est} = \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_2,0} &\equiv 140 < V_{MRSP} < 210 \wedge V_{MRSP} + 5 < V_{est} < \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_2,1} &\equiv V_{MRSP} = 210 \wedge V_{MRSP} + 5 < V_{est} < \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_2,2} &\equiv 140 < V_{MRSP} < 210 \wedge V_{est} = \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_2,3} &\equiv V_{MRSP} = 210 \wedge V_{est} = \frac{209}{200} V_{MRSP} + \frac{55}{100} \\
\Phi_{4_3,0} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 5 < V_{est} < V_{MRSP} + 10 \\
\Phi_{4_3,1} &\equiv 210 < V_{MRSP} \wedge V_{est} = V_{MRSP} + 10
\end{aligned}
$$

$$
\begin{aligned}
\Phi_{5_0,0} &\equiv V_{MRSP} < 110 \wedge V_{MRSP} + 5.5 < V_{est} < V_{MRSP} + 7.5 \\
\Phi_{5_0,1} &\equiv V_{MRSP} = 110 \wedge V_{MRSP} + 5.5 < V_{est} < V_{MRSP} + 7.5 \\
\Phi_{5_0,2} &\equiv V_{MRSP} < 110 \wedge V_{est} = V_{MRSP} + 7.5 \\
\Phi_{5_0,3} &\equiv V_{MRSP} = 110 \wedge V_{est} = V_{MRSP} + 7.5 \\
\Phi_{5_1,0} &\equiv 110 < V_{MRSP} < 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} < \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\Phi_{5_1,1} &\equiv V_{MRSP} = 210 \wedge \frac{209}{200} V_{MRSP} + \frac{55}{100} < V_{est} < \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\Phi_{5_1,2} &\equiv 110 < V_{MRSP} < 210 \wedge V_{est} = \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\Phi_{5_1,3} &\equiv V_{MRSP} = 210 \wedge V_{est} = \frac{43}{40} V_{MRSP} - \frac{3}{4} \\
\Phi_{5_2,0} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 10 < V_{est} < V_{MRSP} + 15 \\
\Phi_{5_2,1} &\equiv 210 < V_{MRSP} \wedge V_{est} = V_{MRSP} + 15 \\
\Phi_{6_0,0} &\equiv V_{MRSP} < 110 \wedge V_{MRSP} + 7.5 < V_{est} \\
\Phi_{6_0,1} &\equiv V_{MRSP} = 110 \wedge V_{MRSP} + 7.5 < V_{est} \\
\Phi_{6_1,0} &\equiv 110 < V_{MRSP} < 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est} \\
\Phi_{6_1,1} &\equiv V_{MRSP} = 210 \wedge \frac{43}{40} V_{MRSP} - \frac{3}{4} < V_{est} \\
\Phi_{6_2} &\equiv 210 < V_{MRSP} \wedge V_{MRSP} + 15 < V_{est}
\end{aligned}
$$

9.5 IECP Refinement by Sub-paving

After having considered all case distinctions related to requirements and all boundary value situations, further refinements are motivated by the possibility of trapdoors implemented in the SUT, so that erroneous behaviour is revealed in certain SUT states for unknown subsets of the input classes obtained so far by the previous two refinement steps.

In principle, trapdoors triggered by a subset containing an interval vector of the input domain with positive diameter can be detected by further refining the input classes using sub-paving, that is, by partitioning the input classes using intersections with interval vectors [16]. In absence of any hints concerning size and location of trapdoors inside the existing input classes, there is no evidence, however, that this systematic partitioning will be more effective than just choosing additional uniformly distributed random values from each input class, in addition to the inputs systematically constructed in the previous steps.

9.6 Effects of IECP Refinements on W-Method Application

Since IECP refinements always increase the input alphabet, but do not refine the quiescent or transient state classes, previous test suites performed for a coarser alphabet can be re-used, if the following rules are applied.

1. The characterisation set W remain unchanged by refinements, since it uniquely identifies states, and states are not refined by IECP refinements.

2. The state transition cover STC is always *extended* in the following way.

Table 12. Extended input alphabet $\mathcal{A}_I$ containing boundary values.

$\vec{c}_i$	V_{MRSP}	V_{est}	allowRevokeEB	X_i	$X_{ij,k}$
$\vec{c}_1$	90	60	0	X_1	$X_{1_0,0}$
$\vec{c}_2$	90	60	1	X_2	$X_{2_1,0}$
$\vec{c}_3$	150	152	0	X_3	$X_{3_2,0}$
$\vec{c}_4$	120	125	1	X_4	$X_{4_1,0}$
$\vec{c}_5$	60	66	0	X_5	$X_{5_0,0}$
$\vec{c}_6$	230	260	1	X_6	X_{6_2}
$\vec{c}_7$	90	0	1	X_2	X_{2_0}
$\vec{c}_8$	90	93	0	X_3	$X_{3_0,0}$
$\vec{c}_9$	112	115	1	X_3	$X_{3_1,0}$
$\vec{c}_{10}$	211	212	0	X_3	$X_{3_3,0}$
$\vec{c}_{11}$	90	95	1	X_4	$X_{4_0,0}$
$\vec{c}_{12}$	150	156	0	X_4	$X_{4_2,0}$
$\vec{c}_{13}$	220	226	1	X_4	$X_{4_3,0}$
$\vec{c}_{14}$	205	215	0	X_5	$X_{5_1,0}$
$\vec{c}_{15}$	230	244	1	X_5	$X_{5_2,0}$
$\vec{c}_{16}$	55	100	0	X_6	$X_{6_0,0}$
$\vec{c}_{17}$	200	215	1	X_6	$X_{6_1,0}$
$\vec{c}_{18}$	90	90	0	X_1	$X_{1_0,1}$
$\vec{c}_{19}$	90	90	1	X_2	$X_{2_1,1}$
$\vec{c}_{20}$	110	113	0	X_3	$X_{3_0,1}$
$\vec{c}_{21}$	90	94	0	X_3	$X_{3_0,2}$
$\vec{c}_{22}$	110	114	0	X_3	$X_{3_0,3}$
$\vec{c}_{23}$	140	143	1	X_3	$X_{3_1,1}$
$\vec{c}_{24}$	120	124,3	1	X_3	$X_{3_1,2}$
$\vec{c}_{25}$	140	145	1	X_3	$X_{3_1,3}$
$\vec{c}_{26}$	210	212	0	X_3	$X_{3_2,1}$
$\vec{c}_{27}$	150	155	0	X_3	$X_{3_2,2}$
$\vec{c}_{28}$	210	215	0	X_3	$X_{3_2,3}$
$\vec{c}_{29}$	211	216	0	X_3	$X_{3_3,1}$
$\vec{c}_{30}$	110	115	1	X_4	$X_{4_0,1}$
$\vec{c}_{31}$	90	95,5	1	X_4	$X_{4_0,2}$
$\vec{c}_{32}$	110	115,5	1	X_4	$X_{4_0,3}$
$\vec{c}_{33}$	140	146	1	X_4	$X_{4_1,1}$
$\vec{c}_{34}$	120	125,95	1	X_4	$X_{4_1,2}$
$\vec{c}_{35}$	140	146,85	1	X_4	$X_{4_1,3}$
$\vec{c}_{36}$	210	218	0	X_4	$X_{4_2,1}$
$\vec{c}_{37}$	150	157,3	0	X_4	$X_{4_2,2}$
$\vec{c}_{38}$	210	220	0	X_4	$X_{4_2,3}$
$\vec{c}_{39}$	220	230	1	X_4	$X_{4_3,1}$
$\vec{c}_{40}$	110	116	0	X_5	$X_{5_0,1}$
$\vec{c}_{41}$	60	67,5	0	X_5	$X_{5_0,2}$
$\vec{c}_{42}$	110	117,5	0	X_5	$X_{5_0,3}$
$\vec{c}_{43}$	210	215	0	X_5	$X_{5_1,1}$
$\vec{c}_{44}$	200	214,25	0	X_5	$X_{5_1,2}$
$\vec{c}_{45}$	210	225	0	X_5	$X_{5_1,3}$
$\vec{c}_{46}$	220	235	1	X_5	$X_{5_2,1}$
$\vec{c}_{47}$	110	118	0	X_6	$X_{6_0,1}$
$\vec{c}_{48}$	210	226	1	X_6	$X_{6_1,1}$

- Whenever a quiescent state class is encountered for the first time during STC construction, all members of the alphabet are applied to the state.

Since this rule was already applied in the initial STC construction, the previous STC is always a subset of the new STC.

Let STC, $\mathcal{A}_I$ denote the state transition cover and input alphabet associated with the old IECP, and let STC', $\mathcal{A}'_I$ denote the ones associated with the refined IECP. Providing that the previous IECP has already been completely tested using the W-Method with test suite $\mathcal{W}(STS)$, the additional test cases to be performed with the refined IECP are

$$\begin{aligned}
\mathcal{W}'(STS) - \mathcal{W}(STS) \quad = \quad & (STC' - STC).\mathcal{A}'^{m_0-n}_I.W \cup \\
& STC.(\mathcal{A}'_I - \mathcal{A}_I).\mathcal{A}'^{m_0-n-1}_I.W \cup \\
& STC.\mathcal{A}_I.(\mathcal{A}'_I - \mathcal{A}_I).\mathcal{A}'^{m_0-n-2}_I.W \cup \\
& STC.\mathcal{A}^2_I.(\mathcal{A}'_I - \mathcal{A}_I).\mathcal{A}'^{m_0-n-3}_I.W \cup \\
& \ldots \\
& STC.\mathcal{A}^{m_0-n-1}_I(\mathcal{A}'_I - \mathcal{A}_I).W
\end{aligned}$$

10 Test Procedures

10.1 Test Automation Tool

The tests for the CSM provided under www.mbt-benchmarks.org have been performed with the model-based testing component RTT-MBT of the test automation tool RT-Tester [23]. RT-Tester supports all test levels from unit testing to system integration testing and provides different functions for manual test procedure development, automated test case, test data and test procedure generation, as well as management functions for large test campaigns. The typical application scope covers (potentially safety-critical) embedded real-time systems involving concurrency, time constraints, discrete control decisions as well as integer and floating point data and calculations. While the tool has been used in industry for about 15 years and has been qualified for avionic, automotive and railway control systems under test according to the standards [26, 15, 3], its MBT functionality is based on more recent extensions that have been validated during the last years in various projects from the transportation domains and are now made available to the public.

10.2 Test Categories

The tests presented for the CSM under www.mbt-benchmarks.org can be structured according to the following coverage strategies.

1. Test procedures aiming at basic state coverage
2. Test procedures aiming at transition coverage
3. Test procedures aiming at MC/DC coverage
4. Test procedures based on user defined test cases, aiming at requirements coverage
5. Test procedures based on input equivalence class partitioning principle described in this technical report. These are described in Section 10.4.

10.3 Tests of Categories 1 — 4

Table 13 summarises the coverage achieved by the generated tests of categories 1 — 4. Each test procedure is duplicated to take into account the different values of the flag `SBAvailable`, except for the test procedure TP-005 that checks the behaviour only when service brakes are available. TP-001-BCS is the test procedure realising the basic state coverage, TP-002-Transitions the transition coverage and TP-003-MC/DC the MC/DC coverage. Requirement-based testing is performed by the Test procedures TP-004 to TP-008. Table 14 defines the requirements associated with each test procedure. TP-004 focuses on the DMI interface. TP-005 provides a test to check the behaviour when the service brake is not available. TP-006 validates the special transition when entering the ceiling speed monitoring with the indication status set to one. TP-007 and TP-008 are the test procedure for testing the revocation and the triggering of DMICmd and TICmd.

TP-002-OnlyCSM_ON, TP-002-OnlyCSM_ON-SNOSB

These two test procedures generated by the tool have allowRevokeEB equals 0 and are of the form specified in Table 15 and Table 16. The two test procedures are able to kill the mutant in Example 1 thanks to the sequence NORMAL → OVERSPEED → NORMAL. The second mutant is not killed by the first test procedure but the sequence NORMAL → OVERSPEED → WARNING in

Table 13. Test procedures

TP Name	MCDC	UD	HITR	TR	BCS	REQ
TP-001-BCS	42%	15%	0%	45%	100%	12%
TP-001-BCS-NOSB	42%	15%	0%	45%	100%	12%
TP-002-OnlyCSM_ON	57%	23%	0%	81%	100%	40%
TP-002-OnlyCSM_ON-NOSB	57%	23%	0%	81%	100%	36%
TP-002-Transition	64%	38%	100%	90%	100%	56%
TP-002-Transition-NOSB	78%	23%	100%	100%	100%	52%
TP-003-MCDC	85%	15%	0%	54%	100%	20%
TP-003-MCDC-NOSB	85%	30%	0%	54%	100%	20%
TP-004-3.13.10-DMIOutputs	85%	46%	100%	90%	100%	80%
TP-004-3.13.10-DMIOutputs-NOSB	85%	61%	100%	90%	100%	80%
TP-005-3.13.10-EBvsSB	64%	30%	20%	90%	100%	48%
TP-006-3.13.10-IndicationStatus	57%	38%	60%	63%	100%	24%
TP-006-3.13.10-IndicationStatus-NOSB	57%	46%	60%	63%	100%	28%
TP-007-3.13.10-RevokeCmd	57%	15%	0%	81%	100%	36%
TP-007-3.13.10-RevokeCmd-NOSB	64%	30%	0%	81%	100%	36%
TP-008-3.13.10-Trigger	85%	38%	100%	90%	100%	76%
TP-008-3.13.10-Trigger-NOSB	85%	69%	100%	90%	100%	84%

Table 14. Requirement-based Test procedures

Test Procedure	Requirements
TP-004	REQ-3.13.10.2.1 REQ-3.13.10.3.1 REQ-3.13.10.3.2
TP-005	REQ-3.13.10.2.3 REQ-3.13.10.2.4 REQ-3.13.10.3.3.r1
TP-006	REQ-3.13.10.3.6 REQ-3.13.10.2.5
TP-007	REQ-3.13.10.2.5 REQ-3.13.10.3.3.r0 REQ-3.13.10.3.3.r1
TP-008	REQ-3.13.10.3.3.t[1-5] REQ-3.13.10.3.4

the second test procedure can. Finally, these two procedures are not able to kill the mutant of Example 3.

Table 15. TP-002-OnlyCSM_ON Test suite

(V_{est}, V_{MRSP})	(192, 169)	(0, 179)	(183, 179)	(183, 231)	(183, 0)	(0, 0)	(4.5, 0)	(4.5, 247)
State	S_BRAKE	NORM	OVSP	NORM	E_BRAKE	NORM	WARN	NORM

Table 16. TP-002-OnlyCSM_ON-NOSB Test suite

(V_{est}, V_{MRSP})	(12.5, 6.5)	(0, 6.5)	(6.9, 6.5)	(6.9, 7)	(6.9, 1.5)	(0, 1.5)	(32, 1.5)	(0, 1.5)
State	S_BRAKE	NORM	OVSP	NORM	WARN	NORM	E_BRAKE	NORM

The following tables summarise the experiments performed with the tests of categories 1 — 4, in order to assess their strength. Mutant 1,2 and 3 refers to the Example 1,2 and 3, as presented section 8. Table 17 shows the results obtained after running each test procedure against the 3 mutants. Whenever the test passed it means that the mutant was *not* detected by the test procedure. However when the test failed, the test procedure could witness the erroneous behaviour and therefore kill the mutant. Table 18 summarises the test sequences that killed the mutants for each test procedure. None of the category 1 — 4 tests could kill mutant 3.

10.4 Tests of Category 5 – IECP Tests

In addition to the tests of categories 1 — 4 described above, an IECP test suite was generated, based on a prototype implementation of the IECP test strategy described in Section 5 in RTT-MBT. This implementation was used to generate IECP test cases for the lower-level state machine CSM_ON modelling the behaviour of the active CSM with the input, output, and internal model variables V_{est}, V_{MRSP}, DMICmd, TICmd, DMIdisplaySBI. The tests were based further on the assumptions $sb_0 = 1$ and allowRevokeEB = 0

Table 17. Mutants experiments results

Test-Procedure	Mutant 1	Mutant 2	Mutant 3
TP-001-BCS	PASS	PASS	PASS
TP-001-BCS-NOSB	PASS	PASS	PASS
TP-002-OnlyCSM_ON	FAIL	PASS	PASS
TP-002-OnlyCSM_ON-NOSB	FAIL	FAIL	PASS
TP-002-Transition	FAIL	FAIL	PASS
TP-002-Transition-NOSB	FAIL	FAIL	PASS
TP-003-MCDC	PASS	PASS	PASS
TP-003-MCDC-NOSB	PASS	PASS	PASS
TP-004-3.13.10-DMIOutput	FAIL	PASS	PASS
TP-004-3.13.10-DMIOutput-NOSB	FAIL	PASS	PASS
TP-005-3.13.10-EBvsSB	FAIL	PASS	PASS
TP-006-3.13.10-IndicationStatus	FAIL	PASS	PASS
TP-006-3.13.10-IndicationStatus-NOSB	FAIL	PASS	PASS
TP-007-3.13.10-RevokeCmd	FAIL	FAIL	PASS
TP-007-3.13.10-RevokeCmd-NOSB	FAIL	FAIL	PASS
TP-008-3.13.10-Trigger	FAIL	FAIL	PASS
TP-008-3.13.10-Trigger-NOSB	FAIL	FAIL	PASS

Table 18. Test sequences that kills the mutant

Mutant	Test Procedure	sequences
Mutant 1	TP-002	NORM → OVSP→NORM
	TP-004	NORM → OVSP→NORM
	TP-005	NORM → OVSP→NORM
	TP-006	NORM → OVSP→NORM
	TP-007	NORM → OVSP→NORM
	TP-008	NORM → OVSP→NORM
Mutant 2	TP-002	OVSP → NORM→OVSP
	TP-005	OVSP→ NORM→ WARN
	TP-007	OVSP→ NORM→ WARN
	TP-008	OVSP→NORM→OVSP

In addition to the IECP test suites introduced in Section 7, a refined test suite based on 66 input equivalence classes was performed; this suite provided full requirements coverage and adds selected boundary value test cases. The generation of input classes, selection of the input alphabet, and the application of the W-Method results in 1630 test cases, assuming no additional states in the implementation ($m = 4$). The generation of the whole test suite took approx. 11 minutes. As expected, the IECP test suites killed the three mutants from Example 1 — 3. For more details readers are referred to www.mbt-benchmarks.org.

11 Related Work

The test method described and illustrated in this technical report is a specific instance of *partition testing* approaches, where the input domains of the SUT are divided into subsets, and small numbers of candidates are chosen from each of these sets [19]. The formalisation of equivalence classes is typically based on a *uniformity hypothesis* as introduced in [9]. In the context of safety-relevant applications, the identification of subsets, as well as the selection of representatives from each set, has to be justified. To this end, heuristic techniques such as the classification tree method described in [11] have been implemented in interactive test automation tools. They enable users to clearly represent the input combinations taken into account in a test suite. In absence of a model for the required SUT behaviour, however, no formal justification of this selection could be performed, since internal dependencies between input variables could not be formally analysed. As a consequence, classes have been constructed on a per-input variable basis; this resulted in partitions $X_{i1}, X_{i2}, X_{i3}, \ldots$, structuring the domain of input variable $x_i \in I, i = 1, \ldots, k$. To increase the confidence into the class selection, heuristics like *strong equivalence class testing* have been applied, where it was tried to cover every possible combination $(X_{1j_1}, \ldots, X_{kj_k})$ of classes, by choosing representatives from each of these vectors. In [20], this approach to equivalence class testing is applied to an ATC (automatic train control) system which comprises functionality similar to the one described in this report, but which is based on a national Italian ATC specification. In order to further increase the confidence into the test strength achieved, a grey-box test strategy is applied instead of "purely black-box", so that a subset of internal communication variables can be observed. In addition, equivalence class partition testing is combined with other techniques, such as robustness tests and worst case tests.

The crucial contributions of model-based testing in this context consists in (1) providing insight into the dependencies between input variables and the expected internal control decisions and data transformations performed by the SUT, and (2) in taking into account the expected internal state of the SUT. Contribution (1) allows us to identify input equivalence class partitions in an automated way and to *formally* justify that these classes are adequate for the verification objectives under consideration. Contribution (2) enables us to create an *exhaustive* finite test suite, by applying the W-Method on a state machine resulting from an equivalence class abstraction of the model. A random sequence of input vector applications according to the strong equivalence class testing heuristics only uncovers a certain error, if the vector has been "accidentally" applied in a SUT state where the error could be uncovered by the vector under consideration. In contrast to this, our strategy "drives" the SUT systematically through its state space, so that – provided that the SUT is a member of the fault domain – it is *guaranteed* that a suitable (state,input vector) combination revealing the error will be applied in the resulting test suite.

In model-based testing, the idea to use data abstraction for the purpose of equivalence class definition has been originally introduced in [10], where the classes are denoted as *hyperstates*, and the concept is applied to testing against abstract state machine models. Our results presented here surpass the findings described in [10] in the following ways: (1) while the authors of [10] introduce the equivalence class partitioning technique for abstract state machines only, our

approach extracts partitions from the models' semantic representation. Therefore an exhaustive equivalence class testing strategy can be elaborated for any formalism whose semantics can be expressed by state transition systems. (2) The authors sketch for white box tests only how an exhaustive test suite could be created [10, Section 4]: the transition cover approach discussed there is only applicable for SUT where the internal state (respectively, its abstraction) can be monitored during test execution. (3) The authors only consider finite input sets whose values have been fixed *a priori* [10, Section 2], whereas our approach allows for inputs from arbitrary domains.

Applications of model-based testing in the railway domain are currently investigated by numerous research groups and enterprises. The paper [7] reports on how a railway signalling manufacturer successfully adopted and applied a two-step approach for the verification of two ATP systems reducing their validation costs by about 70%[4]. In the first step MBT was applied by means of the Simulink/Stateflow platform to test the compliance of the implementation of the ATP system with the requirements, and in the second step abstract interpretation was applied by means of the Polyspace tool in order to detect runtime errors like buffer overflows. The MBT approach includes a form of automated back-to-back testing and an additional evaluation that no unexpected behaviours have been introduced by the model-to-code translation process. In contrast to our fully automated strategy, the test suites are created in a manual way (at least one function for each requirement) and equivalence class testing is not covered in this approach. No formal guarantees with respect to the test suites' completeness have been made.

In [2] the TTCN-3 test language is applied in a case study of interlocking systems testing. The results presented there take into account that in this application field the topology of the railway network has to be considered; these aspects have not been covered by our approach presented in this report, which is more applicable to onboard controllers whose behaviour is fairly independent on a specific network topology.

In [8] the authors describe MBT strategies in a distributed real-time setting. The concepts are applied to speed/distance monitoring for rolling stock trains travelling close to each other on the same track and in the same direction. While their example also involves infinite input domains (distance and speed), the problem of equivalence class selection is not considered. Our approach could be applied "locally" to each of the controllers involved in that case study.

12 Conclusion

In this technical report, a SysML model for the Ceiling Speed Monitor of the ETCS onboard controller has been presented and made publicly available on the website www.mbt-benchmarks.org, for the purpose of testing theory evaluation and MBT tool comparisons. The model has been represented in SysML, and a formal model semantics based on state transition systems has been specified by presenting the model's transition relation in propositional form.

A novel equivalence class testing strategy has been applied to derive tests from the CSM model in an automated way. This strategy allows test suite creation depending on a given fault model and guarantees completeness of the generated suites for all members of the associated fault domain. The evaluation shows that for certain types of mutants, the equivalence class testing strategy is significantly stronger than that of other test strategies, such as model transition coverage or MC/DC coverage.

[4] In [18] the authors report cost reductions of at least 30%.

13 Ongoing and Future Work

The mutations used for the evaluation in this technical report were mainly constructed for illustration purposes. Currently, we are evaluating the test strength of IECP test suites in comparison with other model coverage criteria, using large numbers of mutants created by a random generator that mutates models and creates executable "SUT" code from each mutation. These results will also be published on www.mbt-benchmarks.org.

The test suites described in this technical report focused on the active CSM only. We are currently working on a completion of the ETCS speed supervision functions, elaborating SysML sub-models for target speed monitoring and release speed monitoring. The resulting test suites will then also consider the switching between these three supervision functions.

The CSM test model inspires further investigations with respect to *product line testing* [17]: the system depends on a constant parameter sb_0 marking the availability of a service brake to be used for intervention purposes in case of speed restriction violations (Section 4.5). This parameter is not an input to the SUT, but to be kept constant throughout a test suite, since it refers to the trains' hardware configurations. The SUT behaviour, however, depends on the value of sb_0, so two different test suites have to be produced and exercised on the SUT, one for $sb_0 = 0$, the other for $sb_0 = 1$. This results in the challenge to identify in an automated way which test cases do not depend on sb_0, so that they have to be executed just once, and which cases have to be exercised for every possible value of sb_0.

References

[1] Robert V. Binder. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Addison-Wesley, 2000.

[2] Jens R. Calame, Nicolae Goga, Natalia Ioustinova, and Jaco van de Pol. TTCN-3 Testing of Hoorn-Kersenboogerd Railway Interlocking. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE'06)*, pages 620–623. IEEE, 2006.

[3] CENELEC. *EN 50128:2011 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems*. 2011.

[4] Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, SE-4(3):178–186, March 1978.

[5] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[6] European Railway Agency. ERTMS – System Requirements Specification – UNISIG SUBSET-026, February 2012. Available under http://www.era.europa.eu/Document-Register/Pages/Set-2-System-Requirements-Specification.aspx.

[7] Alessio Ferrari, Gianluca Magnani, Daniele Grasso, Alessandro Fantechi, and Matteo Tempestini. Adoption of Model-based Testing and Abstract Interpretation by a Railway Signalling Manufacturer. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 2(2):42–61, 2011.

[8] Christophe Gaston, RobertM. Hierons, and Pascale Gall. An implementation relation and test framework for timed distributed systems. In Hüsnü Yenigün, Cemal Yilmaz, and Andreas Ulrich, editors, *Testing Software and Systems*, volume 8254 of *Lecture Notes in Computer Science*, pages 82–97. Springer Berlin Heidelberg, 2013.

[9] Marie-Claude Gaudel. Testing can be formal, too. In PeterD. Mosses, Mogens Nielsen, and MichaelI. Schwartzbach, editors, *TAPSOFT '95: Theory and Practice of Software Development*, volume 915 of *Lecture Notes in Computer Science*, pages 82–96. Springer Berlin Heidelberg, 1995.

[10] Wolfgang Grieskamp, Yuri Gurevich, Wolfram Schulte, and Margus Veanes. Generating finite state machines from abstract state machines. *ACM SIGSOFT Software Engineering Notes*, 27(4):112–122, July 2002.

[11] Matthias Grochtmann and Klaus Grimm. Classification Trees for Partition Testing. *Software Testing, Verification and Reliability*, 3(2):63–82, 1993.

[12] Wen-ling Huang and Jan Peleska. Exhaustive model-based equivalence class testing. In Hüsnü Yenigün, Cemal Yilmaz, and Andreas Ulrich, editors, *Testing Software and Systems*, volume 8254 of *Lecture Notes in Computer Science*, pages 49–64. Springer Berlin Heidelberg, 2013.

[13] Wen-ling Huang, Jan Peleska, and Uwe Schulze. Test automation support. Technical Report D34.1, COMPASS Comprehensive Modelling for Advanced Systems of Systems, 2013. Available under `http://www.compass-research.eu/deliverables.html`.

[14] International Organization for Standardization. *ISO 26262 - Road Vehicles - Functional Safety - Part 8: Supporting Processes*. 2009. ICS 43.040.10.

[15] ISO/DIS 26262-4. Road vehicles – functional safety – part 4: Product development: system level. Technical report, International Organization for Standardization, 2009.

[16] Luc Jaulin, Michel Kieffer, Olivier Didrit, and Éric Walter. *Applied Interval Analysis*. Springer-Verlag, London, 2001.

[17] Beatriz Pérez Lamancha, Macario Polo Usaola, and Mario Piattini Velthuis. Software product line testing - a systematic review. In Boris Shishkov, José Cordeiro, and Alpesh Ranchordas, editors, *ICSOFT (1)*, pages 23–30. INSTICC Press, 2009.

[18] Helge Löding and Jan Peleska. Timed moore automata: test data generation and model checking. In *Proc. 3rd International Conference on Software Testing, Verification and Validation (ICST'10)*. IEEE Computer Society, 2010.

[19] Giuseppe Nicola, Pasquale Tommaso, Esposito Rosaria, Flammini Francesco, Marmo Pietro, and Orazzo Antonio. A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems. In Mario Cin, Mohamed Kaâniche, and András Pataricza, editors, *Dependable Computing - EDCC 5*, volume 3463 of *Lecture Notes in Computer Science*, pages 305–317. Springer, 2005.

[20] Giuseppe De Nicola, Pasquale di Tommaso, Rosaria Esposito, Francesco Flammini, and Antonio Orazzo. A Hybrid Testing Methodology for Railway Control Systems. In Maritta Heisel, Peter Liggesmeyer, and Stefan Wittmann, editors, *Proceedings of SAFECOMP 2004*, volume 3219 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2004.

[21] Object Management Group. OMG Systems Modeling Language (OMG SysML^{TM}). Technical report, Object Management Group, 2010. OMG Document Number: formal/2010-06-02.

[22] Object Management Group. OMG Unified Modeling Language (OMG UML), superstructure, version 2.4.1. Technical report, OMG, 2011.

[23] Jan Peleska. Industrial-strength model-based testing - state of the art and current challenges. In Alexander K. Petrenko and Holger Schlingloff, editors, Proceedings Eighth Workshop on *Model-Based Testing,* Rome, Italy, 17th March 2013, volume 111 of *Electronic Proceedings in Theoretical Computer Science*, pages 3–28. Open Publishing Association, 2013.

[24] Jan Peleska, Artur Honisch, Florian Lapschies, Helge Löding, Hermann Schmid, Peer Smuda, Elena Vorobev, and Cornelia Zahlten. A real-world benchmark model for testing concurrent real-time systems in the automotive domain. In Burkhart Wolff and Fatiha Zaidi, editors, *Testing Software and Systems. Proceedings of the 23rd IFIP WG 6.1 International Conference, ICTSS 2011*, volume 7019 of *LNCS*, pages 146–161, Heidelberg Dordrecht London New York, November 2011. IFIP WG 6.1, Springer.

[25] A. Petrenko, N. Yevtushenko, and G. v. Bochmann. *Fault Models for Testing in Context*, pages 163–177. Chapman&Hall, 1996.

[26] RTCA,SC-167. *Software Considerations in Airborne Systems and Equipment Certification, RTCA/DO-178B*. RTCA, 1992.

[27] Andreas Spillner, Tilo Linz, and Hans Schaefer. *Software Testing Foundations*. dpunkt.verlag, Heidelberg, 2006.

[28] UNISIG. *ERTMS/ETCS SystemRequirements Specification, Chapter 3, Principles*, volume Subset-026-3. February 2012. Issue 3.3.0.

[29] M. P. Vasilevskii. Failure diagnosis of automata. *Kibernetika (Transl.)*, 4:98–108, July-August 1973.

[30] RTCA SC-205/EUROCAE WG-71. Software Considerations in Airborne Systems and Equipment Certification. Technical Report RTCA/DO-178C, RTCA Inc, 1140 Connecticut Avenue, N.W., Suite 1020, Washington, D.C. 20036, December 2011.