Complete Model-Based Equivalence Class Testing for the ETCS Ceiling Speed Monitor

Braunstein, Cécile; Haxthausen, Anne Elisabeth; Huang, Wen-ling; Hübner, Felix; Peleska, Jan; Schulze, Uwe; Vu, Linh Hong

Published in:
Formal Methods and Software Engineering

Link to article, DOI:
10.1007/978-3-319-11737-9_25

Publication date:
2014

Document Version
Peer reviewed version

Citation (APA):
Braunstein, C., Haxthausen, A. E., Huang, W., Hübner, F., Peleska, J., Schulze, U., & Vu, L. H. (2014). Complete Model-Based Equivalence Class Testing for the ETCS Ceiling Speed Monitor. In Formal Methods and Software Engineering: Proceedings of the 16th International Conference on Formal Engineering Methods, ICFEM 2014 (pp. 380-395). Springer. (Lecture Notes in Computer Science, Vol. 8829). DOI: 10.1007/978-3-319-11737-9_25

Complete Model-Based Equivalence Class Testing for the ETCS Ceiling Speed Monitor

Cécile Braunstein[1*], Anne E. Haxthausen[3†], Wen-ling Huang[12***], Felix Hübner[1**], Jan Peleska[1***], Uwe Schulze[1***], and Linh Vu Hong[3†]

[1] Department of Mathematics and Computer Science
University of Bremen, Germany
[2] Department of Mathematics
University of Hamburg, Germany
[3] DTU Compute
Technical University of Denmark

Abstract. In this paper we present a new test model written in SysML and an associated blackbox test suite for the Ceiling Speed Monitor (CSM) of the European Train Control System (ETCS). The model is publicly available and intended to serve as a novel benchmark for investigating new testing theories and comparing the capabilities of model-based test automation tools. The CSM application inputs velocity values from a domain which could not be completely enumerated for test purposes with reasonable effort. We therefore apply 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 complete (i.e. sound and exhaustive) for a given fault model. In this paper, an overview of the model and the equivalence class testing strategy is given, and tool-based evaluation results are presented. For the technical details we refer to the published model and a technical report that is also available on the same website.

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

1 Introduction

In 2011 the *model-based testing benchmarks website* www.mbt-benchmarks.org has been created. Its objective is to publish test models that may serve as chal-

* The author's research is funded by ITEA2 project openETCS under grant agreement 11025

** The author's research is funded by Siemens AG in the context of the SyDE Graduate School on System Design, http://www.informatik.uni-bremen.de/syde/index.php?home-en

*** The authors' research is funded by the EU FP7 COMPASS project under grant agreement no.287829.

† The authors' research is funded by the RobustRailS project funded by the Danish Council for Strategic Research.

lenges or benchmarks for validating testing theories and for comparing the capabilities of model-based testing (MBT) tools [12]. In the present paper 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 [4]. In the first part of this paper (Section 2) we give an introduction into the CSM model.

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 test strategies identifying finitely many representatives from the input domains have to be applied when testing the CSM, and in this paper we focus on equivalence class partition (ECP) testing. While ECP testing is 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 [14], 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 5. In the second part of this paper (Section 3) 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 [7]. The present paper illustrates its practical application and presents first evaluation details using a prototype implementation in an existing MBT tool (Section 4). This 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 [13], 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 fault domain is a collection of models whose behaviour may or may not be consistent with 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 at least fail once when tested according to the test suite (the test suite is *exhaustive*).

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 [3], 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 [8] it is shown how the semantics of general SysML models and models of a process algebra are encoded as STS. In this paper we illustrate how this is achieved for the concrete case of the CSM SysML model.

2 CSM Model Description

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 [4]. 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."* [15, 3.13.1.1]. While displaying actual and allowed speed to the train engine driver, the monitoring functions automatically trigger the brakes in case of speed limit violations. Speed and distance monitoring is decomposed into three sub-functions [15, 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)[4]. 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)* enforces speed restrictions applicable while the train brakes to a target, for example, a track section where a significantly lower maximal speed has to be observed. (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 and is allowed to drive at a reduced speed.

The model presented here captures the CSM functionality.

Test Model Semantics. SysML test models are structured using blocks. At the top-level, the model is decomposed into a block representing the SUT and another one representing the test environment (TE); Fig. 1 shows this decomposition for the CSM. Depending on the complexity of the model, blocks can be further decomposed into lower-level block diagrams, until leaf blocks are reached that are associated with behaviour. In our test models this behaviour is specified by sequential hierarchic SysML state machines. Blocks execute concurrently and in a synchronous way, so that transitions of concurrent state machines that are enabled in the same model state execute simultaneously.

[4] In some situations, more than one speed restriction may apply, and then the most restrictive one has to be enforced.

The whole model executes according to the *run-to-completion* semantics defined for state machines. The model is in a *quiescent* (or stable) state, if no transition can be executed without an input change.

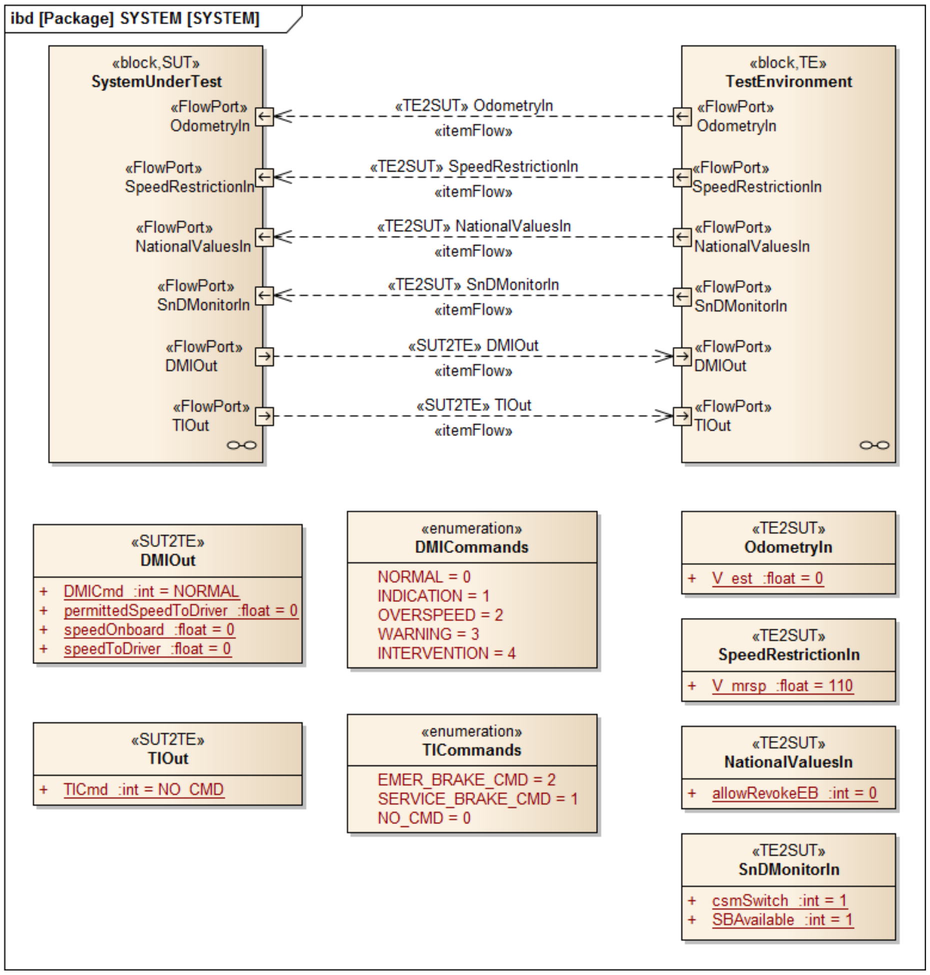

Fig. 1. System interface of the ceiling speed monitor.

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 sequential safety-critical 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.

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.

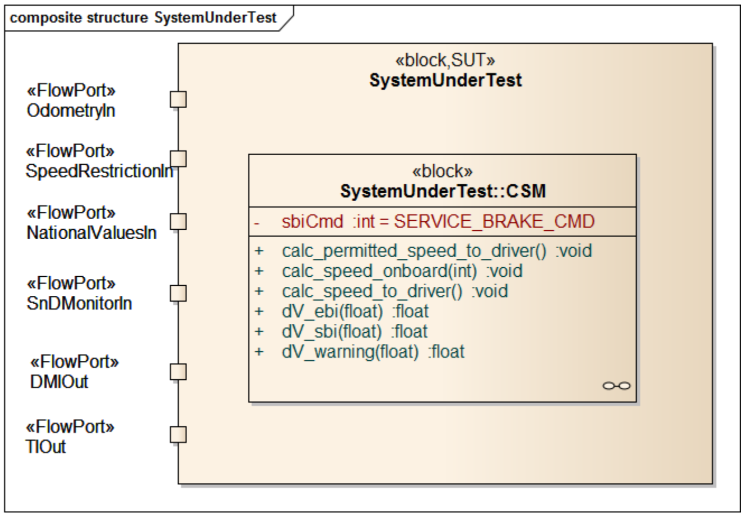
composite structure SystemUnderTest
«block,SUT»
SystemUnderTest
«FlowPort» OdometryIn
«FlowPort» SpeedRestrictionIn
«FlowPort» NationalValuesIn
«FlowPort» SnDMonitorIn
«FlowPort» DMIOut
«FlowPort» TIOut
«block»
SystemUnderTest::CSM
- sbiCmd :int = SERVICE_BRAKE_CMD
+ calc_permitted_speed_to_driver() :void
+ calc_speed_onboard(int) :void
+ calc_speed_to_driver() :void
+ dV_ebi(float) :float
+ dV_sbi(float) :float
+ dV_warning(float) :float

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

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.

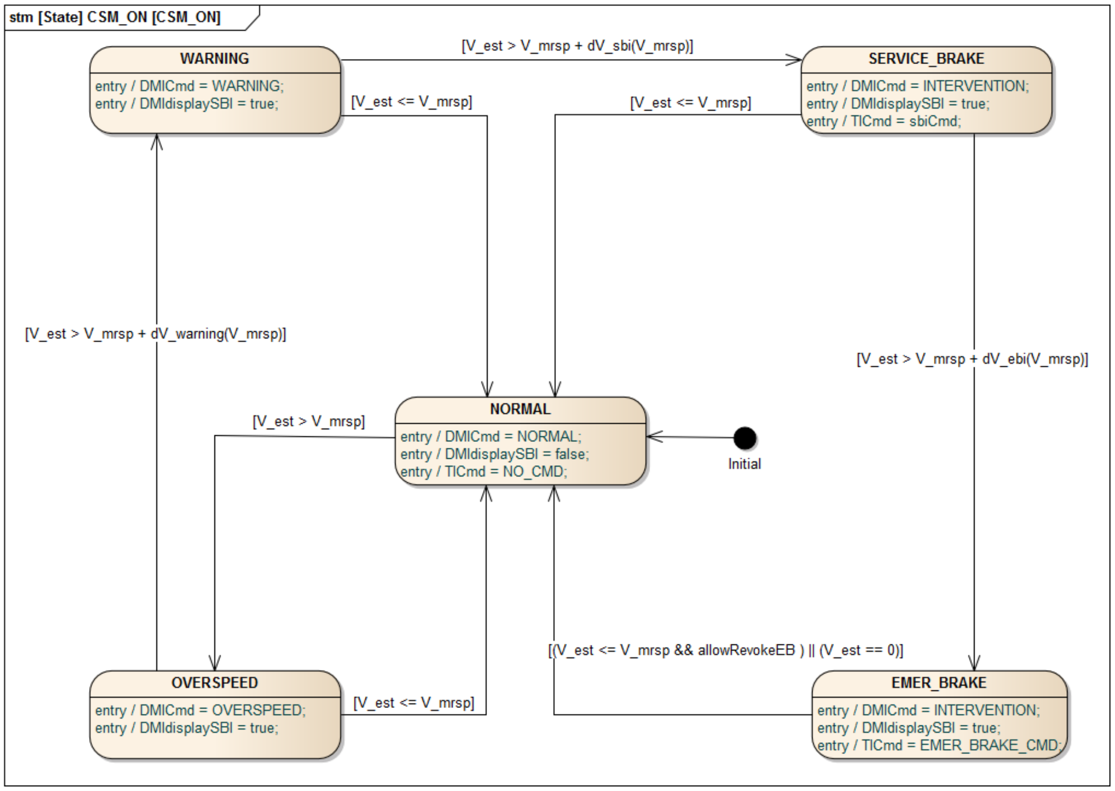
stm [State] CSM_ON [CSM_ON]
WARNING
entry / DMICmd = WARNING;
entry / DMIdisplaySBI = true;
[V_est > V_mrsp + dV_sbi(V_mrsp)]
SERVICE_BRAKE
entry / DMICmd = INTERVENTION;
entry / DMIdisplaySBI = true;
entry / TICmd = sbiCmd;
[V_est <= V_mrsp]
[V_est <= V_mrsp]
[V_est > V_mrsp + dV_warning(V_mrsp)]
[V_est > V_mrsp + dV_ebi(V_mrsp)]
[V_est > V_mrsp]
NORMAL
entry / DMICmd = NORMAL;
entry / DMIdisplaySBI = false;
entry / TICmd = NO_CMD;
Initial
[(V_est <= V_mrsp && allowRevokeEB) || (V_est == 0)]
OVERSPEED
entry / DMICmd = OVERSPEED;
entry / DMIdisplaySBI = true;
[V_est <= V_mrsp]
EMER_BRAKE
entry / DMICmd = INTERVENTION;
entry / DMIdisplaySBI = true;
entry / TICmd = EMER_BRAKE_CMD;

Fig. 3. Ceiling speed monitoring state machine.

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 ($\mathsf{allowRevokeEB} = 0$) 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 several speed-related variables that are displayed on the DMI.

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.

SUT Attributes and Operations. The CSM executes sequentially; therefore the SUT block on the top-level interface diagram (Fig. 1) is refined to a single block representing the CSM, as shown in Fig. 2. 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.

Three supervision limits are computed to assist the driver in preventing automated service or emergency brake intervention by maintaining the speed within certain limits. These limits depend on the MRSP, and they are calculated according to [15] as follows.

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

$$\mathsf{dV}_{\mathsf{sbi}}(V_{MRSP}) = \begin{cases} \min\{0.55 + 0.045 \cdot V_{MRSP}, 10\} & \text{if } V_{MRSP} > 110 \\ 5.5 & \text{if } V_{MRSP} \leq 110 \end{cases} \tag{2}$$

$$\mathsf{dV}_{\mathsf{ebi}}(V_{MRSP}) = \begin{cases} \min\{-0.75 + 0.075 \cdot V_{MRSP}, 15\} & \text{if } V_{MRSP} > 110 \\ 7.5 & \text{if } V_{MRSP} \leq 110 \end{cases} \tag{3}$$

CSM Behavioural Specification. The behaviour of the ceiling speed monitor is modelled by a hierarchic state machine that is associated with the SUT block of Fig. 2. The top-level machine specifies the activation and de-activation of the CSM during the interplay between CSM, TSM, and RSM. Due to the usual space limitations, we consider here only the lower-level state machine CSM_ON modelling the behaviour of the active CSM, as displayed in Fig. 3.

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 $V_{MRSP} + \mathsf{dV}_{\mathsf{warning}}(V_{MRSP})$ is exceeded, a transition into the WARNING state is performed, accompanied by an indication change on the DMI. Accelerating further until $V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{sbi}}(V_{MRSP})$ 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 ($V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{ebi}}(V_{MRSP})$), 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 $V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{ebi}}(V_{MRSP})$ becomes true[5], the state machine transition from NORMAL to OVERSPEED leads to a transient model state, because guard condition $V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{warning}}(V_{MRSP})$ is already fulfilled, and the state machine transits to WARNING. Similarly, guards $V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{sbi}}(V_{MRSP})$ and $V_{est} > V_{MRSP} + \mathsf{dV}_{\mathsf{ebi}}(V_{MRSP})$ also evaluate to true, so that the next quiescent state is reached in basic state EMER_BRAKE.

Full Model Description and Requirements Tracing. The complete SysML model of the CSM function is publicly available[6]. A comprehensive description can be found in the technical report [1] which is also available on this website. The SysML modelling formalism supports the specification of relationships between requirements and model elements contributing to their realisation. This allows for requirements-driven testing: test cases supporting the verification of a given requirement have to cover the model elements contributing to the requirement. In [1] the CSM requirements and the tracing from requirements to model elements, as well as an extended model description are presented.

[5] 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.

[6] `http://www.mbt-benchmarks.org`

3 Equivalence Class Partition Testing Strategy

The theoretical foundations of the equivalence class partition testing method applied in this paper have been described in [7]. In this section we summarise the results obtained there and show how they are applied for testing the CSM.

System Domain. We consider models and SUT whose true behaviour can be represented by state transition systems STS (S, s_0, R) with state space S, initial state $s_0 \in S$ and transition relation $R \subseteq S \times S$. States $s \in S$ are valuation functions $s : V \to D$, where V is a set of variable symbols and $D = \bigcup_{v \in V} D_v$, where D_v is the domain of variable v, and $s(v) \in D_v$ holds for every $v \in V$ and $s \in S$. The variable space V is finite and can be partitioned into disjoint sets $V = I \cup M \cup O$ called input variables, (internal) model variables, and output variables, respectively. The domains of input variables can be infinite, but those of model variables and output variables must be finite. The transition relation $R \subseteq S \times S$ may be infinite, since we allow for infinite input data domains. Admissible STS allow for partitioning of state spaces into quiescent and transient states, $S = S_Q \cup S_T, S_Q \cap S_T = \varnothing$. In a quiescent state $s_1 \in S_Q$ only input changes can occur, leading either to another quiescent, or to a transient post-state s_2. The inputs can then change in an arbitrary way, but the internal and output variables remain unchanged. Transient states $s_1 \in S_T$ have uniquely defined quiescent post-states $s_2 \in S_Q$, and during the transition from s_1 to s_2 only internal variable states and outputs change. The initial state s_0 must be an element of S_Q.

We use initial STS state s_0 to model the quiescent state when "the system is switched off". From there, some input change will drive the STS into the state s the system assumes after initialisation. This state may depend on the new input valuation, so our STS can very well model situations where the initial behaviour depends on the input that is present on system initialisation.

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 $\boldsymbol{x} = (x_1, \ldots, x_k), s(\boldsymbol{x}) = (s(x_1), \ldots, s(x_k))$, $D_I = D_{x_1} \times \cdots \times D_{x_k}$ denotes the cartesian product of the input variable domains. Tuples $\boldsymbol{m}, \boldsymbol{y}$ and D_M and D_O are defined over model variables and outputs in an analogous way. By $s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}\}$, $\boldsymbol{c} \in D_I$ 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.

I/O-Equivalence. Applying a trace $\iota = \boldsymbol{c}_1 \ldots \boldsymbol{c}_n$ of input vectors $\boldsymbol{c}_i \in D_I$ 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 $s_i(\boldsymbol{x}) = \boldsymbol{c}_i, i = 1, \ldots, n$, and $s_i(\boldsymbol{y})$ is the last STS output resulting from application of $\boldsymbol{c}_1 \ldots \boldsymbol{c}_i$ 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 $s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_1\}, \dots, s_{n-1} \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_n\}$, and these states satisfy $R(s \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_1\}, s_1), \dots, R(s_{n-1} \oplus \{\boldsymbol{x} \mapsto \boldsymbol{c}_n\}, s_n)$.

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 $s|_O \neq s'|_O$.

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 $S/_\sim = \{A_1, \dots, A_r\}$. For every $s \in A_i$, applying an input $\boldsymbol{c} \in D_I$ will lead to a quiescent target state denoted by $(s/\!/\boldsymbol{c})$ in the unique target class A_j. Index j only depends on $(i, \boldsymbol{c})$, since for $s, s' \in A_i$ all corresponding states s_k, s'_k in $s/\iota = s_1.s_2 \dots .s_n, s'/\iota = s'_1.s'_2 \dots .s'_n$ are I/O-equivalent for any $\iota = \boldsymbol{c}_1 \dots \boldsymbol{c}_n$, $k = 1, \dots, n$. Therefore $(s/\!/\boldsymbol{c}) \in A_j$ if and only if $(s'/\!/\boldsymbol{c}) \in A_j$. 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 $D_O = \{\boldsymbol{d}_1, \dots, \boldsymbol{d}_{|D_O|}\}$ is finite, we can associate the value index $h \in \{1, \dots, |D_O|\}$ with the target class A_j, if $(s/\!/\boldsymbol{c})|_O = \boldsymbol{d}_h$. Again, h only depends on $(i, \boldsymbol{c})$, but not on the choice of $s \in A_i$.

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

Then the *Input Equivalence Class Partitioning (IECP)* $\mathcal{I} = \{X(\boldsymbol{c}) \mid \boldsymbol{c} \in D_I\}$ 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 $s \in A_i$, $\boldsymbol{c} \in X$, target states $(s/\!/\boldsymbol{c})$ are contained in the same target class $A_{j(i,\boldsymbol{c})}$ and have the unique output value $d_{h(i,\boldsymbol{c})}$. Furthermore, each pair of input traces $\iota = \boldsymbol{c}_1 \dots \boldsymbol{c}_n$, $\iota' = \boldsymbol{c}'_1 \dots \boldsymbol{c}'_n$, when applied to the same state s, lead to the same output traces $(s/\iota)|_O = (s/\iota')|_O$, if $\boldsymbol{c}'_i \in X(\boldsymbol{c}_i)$ for each $i = 1, \dots, n$.

A given IECP $\mathcal{I}$ can be *refined* by selecting input sets $\mathcal{I}_2 = \{X_1, X_2, \dots\}$ 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.

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
$$\forall X \in \mathcal{I}, X' \in \mathcal{I}' : \big(X \cap X' \neq \varnothing \Rightarrow \exists X_2 \in \mathcal{I}_2 : X_2 \subseteq X \cap X'\big)$$
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 reasonable, 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 intuition behind requirement 5 is as follows: for every $X \in \mathcal{I}$ the model $\mathcal{S}$ exhibits equivalent behaviour for every input from X. Non-conforming members $\mathcal{S}'$ of the fault domain may have a different partitioning $\mathcal{I}' \neq \mathcal{I}$. Then there will be some non-empty intersections $X \cap X' \neq \varnothing, X' \in \mathcal{I}'$ that contain inputs for which $\mathcal{S}$ and $\mathcal{S}'$ exhibit different behaviour. It is ensured by requirement 5 that our refined partitioning $\mathcal{I}_2$ has a member X_2 contained in this intersection. This guarantees that an input from $X \cap X'$ will be applied in the test suite introduced below.

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

Complete Test Strategy. The main result of the paper [7] 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 $\boldsymbol{c}_X \in X$ from each $X \in \mathcal{I}_2$.

2. Abstract $\mathcal{S}$ to a finite deterministic state machine $\mathcal{M}$ with I/O-equivalence classes $A_1, \dots, A_r$ as states, input alphabet $\{\boldsymbol{c}_X \mid X \in \mathcal{I}_2\}$ 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 [16, 2] 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 $\mathcal{M}$.

4 Evaluation

The coarsest IECP $\mathcal{I}$ for the CSM model has 6 IECs $X_1, \dots, X_6$; their defining conditions over the input variables are displayed in Table 1. This table also shows the input alphabet, consisting of one input vector selected from each class. It can be easily checked that in a given CSM model state, all inputs from a given X_i lead to the same outputs and into I/O-equivalent quiescent target states.

Table 1. Input Alphabet $\mathcal{A}_I$.

$\boldsymbol{c}_i$	V_{est}	V_{MRSP}	allowRevokeEB	X_i	specified by
$\boldsymbol{c}_1$	60	90	0	X_1	$0 < V_{est} \leq V_{MRSP} \land \mathsf{allowRevokeEB} = 0$
$\boldsymbol{c}_2$	60	90	1	X_2	$V_{est} = 0 \lor (V_{est} \leq V_{MRSP} \land \mathsf{allowRevokeEB} = 1)$
$\boldsymbol{c}_3$	152	150	0	X_3	$V_{MRSP} < V_{est} \leq V_{MRSP} + \mathsf{dV}_{\mathsf{warning}}(V_{MRSP})$
$\boldsymbol{c}_4$	125	120	1	X_4	$V_{MRSP} + \mathsf{dV}_{\mathsf{warning}}(V_{MRSP}) < V_{est} \leq V_{MRSP} + \mathsf{dV}_{\mathsf{sbi}}(V_{MRSP})$
$\boldsymbol{c}_5$	66	60	0	X_5	$V_{MRSP} + \mathsf{dV}_{\mathsf{sbi}}(V_{MRSP}) < V_{est} \leq V_{MRSP} + \mathsf{dV}_{\mathsf{ebi}}(V_{MRSP})$
$\boldsymbol{c}_6$	260	230	0	X_6	$V_{MRSP} + \mathsf{dV}_{\mathsf{ebi}}(V_{MRSP}) < V_{est}$

The DFSM used for test suite generation according to the W-method is shown in Fig. 4. The complete test suites for the fault domain $\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$ derived by application of the W-method are shown in Table 2. The specification of $m = 6$ implies that the domain contains all models whose minimised DFSM representation contains at most two more states than that of the reference model, as shown in Fig. 4.

The tool-based evaluation has been performed with RT-Tester, an industrial strength MBT tool [11] which has been enhanced by a prototype extension supporting IECP test generation as described above. This tool encodes test objectives as propositional formulas, and an SMT solver calculates solutions from which the concrete test data, i.e., the input vectors to the SUT, can be extracted. RT-Tester has been used to generate various test suites from the CSM

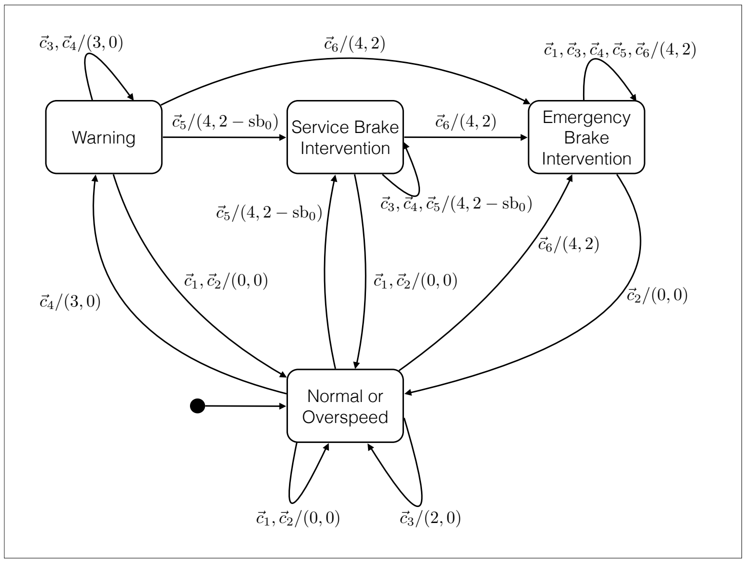
$\vec{c}_3, \vec{c}_4/(3,0)$
$\vec{c}_6/(4,2)$
$\vec{c}_1, \vec{c}_3, \vec{c}_4, \vec{c}_5, \vec{c}_6/(4,2)$
Warning
$\vec{c}_5/(4, 2 - \mathrm{sb}_0)$
Service Brake Intervention
$\vec{c}_6/(4,2)$
Emergency Brake Intervention
$\vec{c}_3, \vec{c}_4, \vec{c}_5/(4, 2 - \mathrm{sb}_0)$
$\vec{c}_5/(4, 2 - \mathrm{sb}_0)$
$\vec{c}_6/(4,2)$
$\vec{c}_1, \vec{c}_2/(0,0)$
$\vec{c}_1, \vec{c}_2/(0,0)$
$\vec{c}_4/(3,0)$
$\vec{c}_2/(0,0)$
Normal or Overspeed
$\vec{c}_1, \vec{c}_2/(0,0)$
$\vec{c}_3/(2,0)$

Fig. 4. DFSM abstraction of the CSM. Output assignment actions $(\mathsf{DMICmd}, \mathsf{TICmd}) = (\alpha, \beta)$ are written as (α, β). The DMICmd are written as 0 for no indication, 2 for overspeed indication, 3 for warning, and 4 for intervention indication. The TICmd are written as 0 for brakes released, 1 for service brake triggered, and 2 for emergency brake triggered.

model, following different coverage criteria: (1) basic state coverage, (2) transition coverage, (3) MC/DC coverage, (4) hierarchic transition coverage for an extended CSM model version including the activation and deactivation of the CSM, (5) requirements-driven test cases, constructed from the links from ETCS requirements to model elements, (6) the IECP test suites shown in Table 2, and (7) a more detailed IECP test suite based on a refinement $\mathcal{I}_2$ of $\mathcal{I}$ with 69 input equivalence classes, leading to 273 test cases: Usage of the coarsest IECP $\mathcal{I}_2 = \mathcal{I}$ specified in Table 1 is adequate if a fault domain is applicable, where all representatives use the same guard conditions as the reference model. Then conformity violations can only occur in output calculations, but never in control decisions. Refined IECPs are necessary, as soon as potential errors in guard conditions have to be taken into account.

Test strategies (1) — (5) uncover at most 2 out of three mutants by "accidentally" using input data revealing the deviations from the reference models: the nature of the mutants was such that none of these strategies can guarantee to find the mutations used. None of these strategies are able to uncover the third mutation, not even the test suite (5) which yields 100% requirements coverage (see [1, Table 17] for a more detailed specification of the mutants). As expected, test suites (6) and (7) kill all three mutants. For all test suites (1) — (7) the

Table 2. Complete test suite for $\mathcal{D}(\mathcal{S}, m = 6, \mathcal{I}_2 = \mathcal{I})$. $\text{TEST\_SUITE}_{\mathsf{sb}_0=1}$ applies to the case where trains are equipped with a separate service brake. $\text{TEST\_SUITE}_{\mathsf{sb}_0=0}$ applies to train configurations where no separate service brake is available, so that only the emergency brake is used.

$$\begin{aligned}
\text{TEST\_SUITE}_{\mathsf{sb}_0=1} &= \{\boldsymbol{c}_i.\boldsymbol{c}_j.\boldsymbol{c}_k.\boldsymbol{c}_3 \mid i,j,k = 1,\dots,6\} \cup \\
&\quad\; \{\boldsymbol{c}_j.\boldsymbol{c}_i.\boldsymbol{c}_k.\boldsymbol{c}_h.\boldsymbol{c}_3 \mid h,i,k = 1,\dots,6, \quad j = 4,\dots,6\} \\
\text{TEST\_SUITE}_{\mathsf{sb}_0=0} &= \{\boldsymbol{c}_i.\boldsymbol{c}_j.\boldsymbol{c}_h.\boldsymbol{c}_g \mid h,i,j = 1,\dots,6, \; g = 1,3\} \cup \\
&\quad\; \{\boldsymbol{c}_j.\boldsymbol{c}_i.\boldsymbol{c}_k.\boldsymbol{c}_h.\boldsymbol{c}_g \mid h,i,k = 1,\dots,6, \quad j = 4,\dots,6, \; g = 1,3\}
\end{aligned}$$

automated test suite generation time is below 60 seconds. These results are summarised in Table 3.

Table 3. Experimental results (see www.mbt-benchmarks.org for more details)

Test-Procedure	Mutant 1	Mutant 2	Mutant 3	Generation Time [s]
Strategy (1)	not detected	not detected	not detected	≤ 60
Strategy (2)	KILLED	KILLED	not detected	≤ 60
Strategy (3)	not detected	not detected	not detected	≤ 60
Strategy (4)	KILLED	KILLED	not detected	≤ 60
Strategy (5)	KILLED	KILLED	not detected	≤ 60
IECP Strategy (6)	KILLED	KILLED	KILLED	≤ 60
IECP Strategy (7)	KILLED	KILLED	KILLED	≤ 60

5 Related Work

The test method described and illustrated in this paper 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 [9]. The formalisation of equivalence classes is typically based on a *uniformity hypothesis* as introduced in [5]. The idea to use data abstraction for the purpose of equivalence class definition has been originally introduced in [6], where the classes are denoted as *hyperstates*, and the concept is applied to testing against abstract state machine models. Complete test suites have been suggested there for grey box scenarios, while our approach considers black-box tests.

Applications of model-based testing in the railway domain are currently investigated by numerous research groups and enterprises. In [1, Section 12] several references are given, and also alternative approaches to tool support are discussed.

The detailed formal behavioural semantics of general SysML test models has been described in [8, pp. 88]. This semantics is consistent with the standard [10], but fixes certain semantic variation points in ways that are admissible according to the standards. In [1, Section 4], the formal semantics of the CSM is presented by specifying the model's transition relation in propositional form. Furthermore, additional details are presented for the IECP $\mathcal{I}$ introduced above [1, Section 5], and IECP refinement alternatives $\mathcal{I}_2$ are discussed [1, Section 6, 10].

6 Conclusion and Ongoing Work

In this paper, a SysML model for the Ceiling Speed Monitor of the ETCS on-board 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. 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.

The usage of SysML was motivated by the fact that this modelling language is very well accepted in industrial applications. It is therefore one of the main modelling formalisms used in the ITEA2 project openETCS[7] and the FP7 project COMPASS[8].

The mutations used for the evaluation in this paper were mainly constructed for illustration purposes. Currently, we are evaluating the test strength of IECP test suites in comparison with other model coverage criteria with 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.

References

1. Cécile Braunstein, Wen-ling Huang, Jan Peleska, Uwe Schulze, Felix Hübner, Anne E. Haxthausen, and Linh Vu Hong. A SysML test model and test suite for the ETCS ceiling speed monitor. Technical report, Embedded Systems Testing Benchmarks Site, 2014-04-30. Available under `http://www.mbt-benchmarks.org`.
2. Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, SE-4(3):178–186, March 1978.
3. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.
4. 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] http://openetcs.org

[8] http://www.compass-research.eu

5. 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.
6. 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.
7. 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.
8. 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`.
9. 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.
10. Object Management Group. OMG Systems Modeling Language (OMG SysML^{TM}). Technical report, Object Management Group, 2010. OMG Document Number: formal/2010-06-02.
11. 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.
12. 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.
13. A. Petrenko, N. Yevtushenko, and G. v. Bochmann. *Fault Models for Testing in Context*, pages 163–177. Chapman&Hall, 1996.
14. Andreas Spillner, Tilo Linz, and Hans Schaefer. *Software Testing Foundations.* dpunkt.verlag, Heidelberg, 2006.
15. UNISIG. *ERTMS/ETCS SystemRequirements Specification, Chapter 3, Principles*, volume Subset-026-3. 2012. Issue 3.3.0.
16. M. P. Vasilevskii. Failure diagnosis of automata. *Kibernetika (Transl.)*, 4:98–108, July-August 1973.