Proc. of the Intl. Conf. on Advances In Information Processing And Communication Technology - IPCT 2014.

ISBN: 978-1-63248-021-7 doi: 10.15224/ 978-1-63248-021-7-91

Verifying and testing ETCS Train Implementations based on IF specifications

Natalia Kushik, Denisa Ianculescu, Ana Cavalli, Nina Yevtushenko, and Mounir Lallali

***Abstract*—This paper presents test generation scenarios for a train implementation based on the requirements for European Train Control System (ETCS). The formal model used for the test derivation is the model of a Timed Extended Finite State Machine (TEFSM) given in the IF language. This language allows to capture some important properties such as safety properties that should be checked for train implementations represented as corresponding test objectives. The tool TestGen-IF is then used for automatic generation of test cases.**

***Keywords*—Timed Extended FSM, ETCS, IF language, TestGen-IF tool.**

I. Introduction

Over the past years formal models for verifying and testing components of European Train Control System (ETCS) are actively discussed in the literature [see, for example, 1, 2], since random and boundary testing is not enough when the safety aspects are involved. In the paper [2], the authors consider the model of a Finite State Machine augmented with context variables, input/output parameters and timeouts, and study the abilities of this model for testing and verification of ETCS components. There are various methods for deriving tests w.r.t. such formal model [see, for example, 3]. In this paper, we discuss in details how such tests can be derived based on the train specification given in the IF language [4]. Similar to [2], we assume that the system has three components: the Radio Block Center (*RBC*), the train, and the environment itself, and we derive tests for checking safety properties of a train implementation.

In this paper, we use TestGen-IF tool [5] to automatically obtain a set of tests for train implementations; the tool algorithms are based on the Hit-or-Jump method [6]. Tests are generated based on test objectives which describe safety properties.

Natalia Kushik, Nina Yevtushenko
Tomsk State University
Russia

Ana Cavalli, Denisa Ianculescu
Telecom SudParis - CNRS SAMOVAR
France

Mounir Lallali
Université Européenne de Bretagne, Brest, France - Lab-STICC, UBO
France

The IF description is first verified by producing expected outputs to all test cases according to the plain description [7] and then the expressiveness of derived tests is evaluated by mutation testing of a Java implementation [2]. The main contributions of this paper are the IF specification of the TEFSM simulating the train behavior in the ETCS, the set of corresponding test objectives for checking safety properties, and the set of tests generated by the TestGen-IF tool. The experimental results [2] show that the set of generated test cases has a good coverage of safety properties. Therefore, those tests can be used for checking other train simulators/implementations in order to estimate their quality.

The rest of the paper is organized as follows. Section II briefly introduces the train formal model. Section III presents the IF language and the TestGen-IF tool, while Section IV contains the brief description of some test objectives and corresponding test cases. Section V concludes the paper. We also mention that the preliminary version of test objectives and some test suites have been presented in the master thesis [8] (Telecom SudParis, Evry, France).

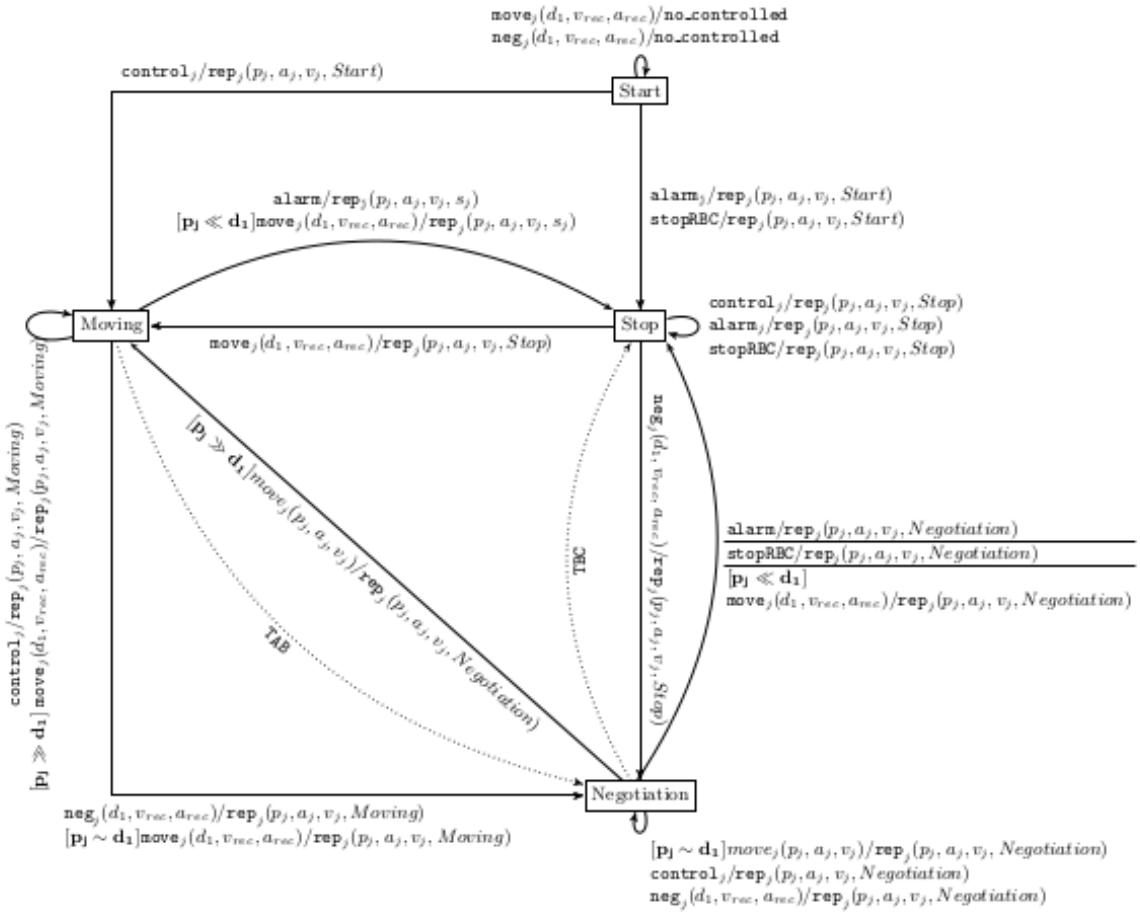
II. Train formal model

The formal model of a train is an Extended FSM with timeouts [2]. In this section, we provide a brief description of this model; the details of the model can be found in [2]. The EFSM has four states which are shown in Figure 1. The input set is the set {*move* (d, V_{max}), *neg* (d, V_{max}), *stopRBC*, *control*(k), *alarm*}, while the output set is {*no_controlled*, *rep*(p, v, a, *current_state*)}. Input parameters d and V_{rec} correspond to a safety distance d; before reaching a critical point w.r.t. the safety distance d, the train can move autonomously with the recommended speed V_{rec}. Output parameters p, v, a, *current_state* are output parameters representing the current train position, the train speed, acceleration, and current state which are mentioned to the RBC that controls the train. Once the train is controlled by the RBC it moves to a corresponding state depending on the safe distance d. For example, when the train crosses the *Negotiation* point the train and the RBC start negotiating, i.e., the train moves to the *Negotiation* state.

If the train should be stopped then it moves to the *Stop* state. The train can be stopped if timeouts are triggered or the current position of the train is very close to the critical point. Another critical stopping situation occurs when the train receives an *alarm* message or a *stopRBC* message from the RBC. In other words, when the train gets the *alarm* or *stopRBC* messages it should be immediately stopped, i.e., the train should move to the *Stop* state from any state. Otherwise, the train continues moving and is at the *Moving* or *Negotiation* state depending on the value of d and the current position of the train.

Proc. of the Intl. Conf. on Advances In Information Processing And Communication Technology - IPCT 2014.

ISBN: 978-1-63248-021-7 doi: 10.15224/ 978-1-63248-021-7-91



$$TAB = 2 \cdot SD/V_{max}$$
$$TBC = 2 \cdot SD/V_{max}$$

Figure 1. TEFSM train specification.

There is a timeout TAB from the *Moving* state to the *Negotiation* state. If the train is in the *Moving* state and the train does not receive any input during TAB time units, then the train automatically moves to the *Negotiation* state. There also is a timeout TBC from the *Negotiation* state to the *Stop* state. If the train is in the *Negotiation* state and the train does not receive any input during TBC time units, then the train automatically moves to the *Stop* state.

III. IF language and TestGen-IF tool

The formal model in Figure 1 has been presented in the IF language [4] where safety (critical) properties can be described as test objectives; test scenarios for checking those properties can be automatically generated using the TestGen-IF tool [5].

A. *IF Specification*

IF is an expressive language that allows to formally describe the existing specification concepts and manage data, time and parallelism. It is based on temporized asynchronously communicating machines that are extended with discrete data, and offers a high support for the formal description and validation of asynchronous systems.

The IF specification of a system consists of active processes running in parallel and interacting asynchronously, communication elements (signals and communication routes), the behavior description of the system in terms of states, transitions, actions which can be separated into inputs and

Proc. of the Intl. Conf. on Advances In Information Processing And Communication Technology - IPCT 2014.

ISBN: 978-1-63248-021-7 doi: 10.15224/ 978-1-63248-021-7-91

outputs, and data variables. Each IF process is described as a timed automaton/FSM extended with communication primitives, data variables and urgency attributes related to transitions. During the IF specification execution, a process instance can be created and destroyed dynamically.

The IF allows to specify real-time systems together with their parallel and asynchronous interactions. Correspondingly, the IF description of a given system contains the definition of: data types, constants, shared variables, communication signals and processes. In Figure 2, a fragment of the IF train description is presented.

```
system ETCS;

/* Constant definitions */
const SD = 1;
const NbTrains = 2;

/* Type definitions */
type states = enum
  start, moving, _stop, negotiation
endenum;
...

/* Signals definitions */
signal ETCS_no_controlled(pid);
signal ETCS_report(pid, integer, integer, integer, states);
signal ETCS_control();
signal ETCS_move(dType, VmaxType, AmaxType);
...

/*Signal route definitions */
signalroute train_to_RBCenv(1)
    from Train to env
    with ETCS_no_controlled, ETCS_report;
....
```

```
/* Main process */
 process Train(2);

 /* Local variables */
    var Vmax integer private;
    var c_TAB clock private;
...

/* States specification */
state start #start ;
    deadline lazy;
    input ETCS_move(d, Vmax, Amax);
        output ETCS_no_controlled(self);
        nextstate -;
    ...
endstate;
...
endprocess;
endsystem;
```

Figure 2. A fragment of the IF code for the train specification.

B. *TestGen-IF tool*

The TestGen-IF tool has been developed by a research team at Telecom SudParis [5] for modeling and testing asynchronous timed systems such as telecommunication protocols and/or distributed applications. The TestGen-IF is based on active testing techniques, allowing automatic timed test case generation from the IF specification of a system under test. Test cases are derived according to given timed test objectives (or test purposes).

The TestGen-IF tool is based on the IF simulator which is a part of IF exploration platform. It allows to construct a partial reachability graph on-the-fly based on the IF specification and guided by test objectives. These objectives describe the functionalities of an implementation under test that we want to check.

TestGen-IF implements a timed test case generation algorithm by adapting a Hit-or-Jump test derivation strategy [6] to IF timed automata. It is written in C++ code, the same implementation language as the IF simulator. Therefore, the inputs necessary for TestGen-IF tool are the formal system specification and the specification of test objectives.

IV. Safety properties and test derivation

The system specification and implementation have to exhibit appropriate safety properties and such properties are described through test objectives. A test objective presents a particular functionality of a given system related to safety issues which can be observed by a test engineer. Once a test objective is described in IF language, it is used for guiding the space exploration of the system states. A test objective is described as a conjunction of conditions and it optionally includes a process instance with the identifier, a state of the system (a source state or a destination state), an action of the system (a message sent, a message received, an internal action), a process variable or a value of a clock variable of the process together with its state (active or inactive).

For testing the ETCS components, many possible scenarios could be taken into consideration. The main focus is however on testing the safety properties, as this is a crucial aspect of the railway transportation system. Moreover, in this case, test purposes are somehow natural as the scenarios used for testing safety properties are inspired from real situations.

The first, very primitive but necessary property relates to the system capability to stop when the system receives an *alarm* message. Independently of a current state of the train, the property has to be exhibited, i.e., at any state the train must stop when receiving an *alarm* as an input. Several input data for this scenario were considered, such as different values for speed, acceleration, as well as the safety distance value that is a constant value in our study. Four set of test sequences were generated, considering each time a new state of the train where an alarm message has been got.

Another intuitive test scenario represents the situation that caused the train accident in Spain in 2013 [9]. The property to be verified is related to the capability of a driver to move too fast (to speed up) in a dangerous area (when approaching a curve, for example). In order to check this property, the RBC should take the control of the train, i.e., the train cannot move autonomously when the critical point is too close to a current train position. If a driver speeds up more than it is allowed by the RBC that controls the train, then after appropriate number of reports, the train should be directed to the *Stop* or to the *Negotiation* state.

Similar to the previous intuitive test scenarios, another scenario can be considered taking into account a critical distance for the train. The critical distance is calculated according to the distance to the previous train, or to some obstacles or according to some other reasons. Depending on the safety distance, a current position, a speed and an acceleration, the train must either change the *Moving* state to the *Stop* state or the *Negotiation* state to the *Moving* state.

Moreover, the train-RBC communication is critical w.r.t. the time instances. If messages between the train and the RBC can be lost, the train should take appropriate decisions after a certain number of time instances and the proposed decisions need to be thouroughly tested.

Proc. of the Intl. Conf. on Advances In Information Processing And Communication Technology - IPCT 2014.

ISBN: 978-1-63248-021-7 doi: 10.15224/ 978-1-63248-021-7-91

We have described some safety properties similar to those described above, in the IF language. In order to derive corresponding test sequences, we have executed the TestGen-IF tool against the train IF specification. Below we provide an example of the test objective related to the *alarm* input message and illustrate the corresponding IF test objective specification.

The first test objective is as follows

OBJ(1)*: The train is in the Moving state, running at 120 km/h. Due to some external unexpected reasons, the train must be stopped as soon as possible and an alarm from the RBC is sent to the train. Automatically, the train should stop and provide a reporting message with its current parameters (state, position...) to the RBC.*

The IF specification of this property is presented in Figure 3.

```
OBJ(1) = OBJ(ord) = {obj1, obj2}
obj1 = cond1 ∧ cond2 ∧ cond3 ∧ cond4
    obj1 = cond1 ∧ cond2 ∧ cond3 ∧ cond4
    cond1 = process: instance = {train}0
    cond2 = state :source: moving
    cond3 = state: destination:stop
    cond4 = action: input alarm_to_train()

obj2= cond1 ∧ cond2 ∧ cond3 ∧ cond4
    cond1 = process: instance = {train}0;
    cond2 = state: source: moving
    cond3 = state: destination:stop
    cond4 = action: output report(train_id,p,v,a,s)
```

Figure 3. A fragment of the IF code for the test objective *OBJ*(1).

According to the system specification, the property *OBJ*(1) has to be held at any state of the train. Therefore, at any state, the train should stop when receiving an *alarm* input. We have derived a test case based on this property for each train state. In Figure 4 the test scenario for the *Moving* state is presented.

```
?ETCS_control{}!ETCS_report{{Train}0,10,100,10,start}
?ETCS_alarm_to_train{}!ETCS_report{{Train}0,10,100,10,moving}
?ETCS_control{}!ETCS_report{{Train}0,10,0,0,_stop}
```

Figure 4: A fragment of the IF code for the test objective *OBJ*(1)

To sum it up, based on the IF train specification (Figure 1) a number of safety properties have been described. Based on the IF description of safety properties, we have derived tests for checking critical railway situations. Tests which were generated based on test objectives have been executed against a Java train simulator and the performed experiments show that the fault coverage of derived test suites is rather high w.r.t. to Java mutation operators [2].

However, test objectives can be more complex and corresponding test cases will be derived using more complicated procedures. For example, another test objective can be considered.

OBJ(2)*: The train is in the Moving state, running at 120 km/h while the permissible speed is 100 km/h. If after the second message with the recommended speed the train speed still exceeds 100 km/h more than 15 km/h the train has to be stopped.*

One of the possibilities to derive a test suite w.r.t. more complex test objectives is to mentally unfold a TEFSM to a classical FSM with the limited number of states/transitions and based on the given TEFSM, to derive a test suite that would traverse many transitions of the unfolded FSM. The reason is that a transition tour for a classical FSM detects all possible output implementation faults. Moreover, it is experimentally shown that for an extended FSM, a transition tour performed for the corresponding unfolded FSM has a high fault coverage [10].

We have generated a set of test sequences based on the train model in Figure 1 that will traverse many transitions in the corresponding unfolded FSM. Since inputs are parameterized, we have fixed some of them and the others have been incrementally increased/decreased taking into account physical reasons and dependencies. Correspondingly, one may expect that test sequences for which a test objective to traverse many parameterized transitions in the specification TEFSM is added, should have a high fault coverage.

v. Concluding Remarks

In this paper, we have illustrated how the ETCS components can be tested and verified w.r.t. critical safety properties using the IF language and corresponding software tools. Safety properties have been described as IF test objectives and the TestGen-IF tool has been used for deriving corresponding test cases. The expressiveness and fault coverage of the derived set of test sequences have been experimentally shown to be high and thus, this set can be used to check various train simulators and implementations.

References

[1] J. Peleska, “Industrial-strength model-based testing - state of the art and current challenges,” in 8th Workshop on Model-Based Testing, MBT’13, 2013, pp. 3–28.

[2] C. Andrés, A. Cavalli, N. Yevtushenko, J. Santos, and R. Abreu, “On Modeling and Testing Components of the European Train Control System,” The International Conference on Advances in Information Processing and Communication Technology, 2014 (accepted for publication).

[3] M. G. Merayo, M. Núñez, and I. Rodrguez, “Formal testing from timed finite state machines,” Computer Networks, vol. 52, no. 2, pp. 432–460, 2008.

[4] M. Bozga, J.-C. Fernandez, L. Ghirvu, S. Graf, J.-P. Krimm, L. Mounier, “If: An Intermediate Representation and Validation Environment for Timed Asynchronous Systems,” FM’99 — Formal Methods, Lecture Notes in Computer Science, vol. 1708, pp. 307-327, 1999.

[5] A.R. Cavalli, E. Montes De Oca, W. Mallouli, M. Lallali, “Two Complementary Tools for the Formal Testing of Distributed Systems with Time Constraints,” The 12-th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, 2008.

Proc. of the Intl. Conf. on Advances In Information Processing And Communication Technology - IPCT 2014.

ISBN: 978-1-63248-021-7 doi: 10.15224/ 978-1-63248-021-7-91

[6] A. Cavalli, D. Lee, C. Rinderknecht, and F. Zaidi, "Hit-or-Jump: An algorithm for embedded testing with applications to in services," in Formal Methods for Protocol Engineering and Distributed Systems, ser. IFIP Advances in Information and Communication Technology. Springer, 1999, vol. 28, pp. 41–56.

[7] European Research Agency, "ERTMS/ETCS, functional requirements specification," European Union, http://www.era.europa.eu/Document-Register/Pages/ERA-ERTMS-03204.aspx, Brussels, Belgium, 2007.

[8] Denisa Ianculescu, "Modeling and validation of embedded systems. Application to the European Train Control System," The Master thesis, 2014, p. 45.

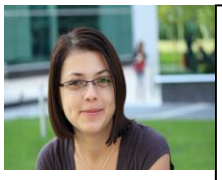
[9] CNN news, http://edition.cnn.com/2013/07/30/world/europe/spain-train-crash/, July 30, 2013.

[10] Natalia Kushik, Anton Kolomeez, Ana Cavalli, and Nina Yevtushenko, "Extended Finite State Machine based Test Derivation Strategies for Telecommunication Protocols," The Spring/Summer Young Researchers' Colloquium on Software Engineering, 2014 (accepted for publication).

Acknowledgment

The work is partially supported by RFBR grant № 14-08-31640 мол_а (Russia).

Natalia Kushik has received her diploma degree in Applied Mathematics from Tomsk State University, Russia, in 2010. She has got a PhD degree in 2013 and worked as a PostDoc at Telecom SudParis, France. Her research interests include automata theory, service quality evaluation, software testing and verification.



Denisa Ianculescu has received her master degree in 2012 at The Polytechnic University of Bucharest. In 2013 she completed her second master thesis in the TELECOM SudParis working on analyzing safety properties of the ETCS components.

Ana Rosa Cavalli is Full Professor at TELECOM SudParis since 1990, and nowadays she is the director of the Software for Networks department. Her research interests are on formal modeling, testing methodologies for conformance and interoperability testing, active testing and monitoring techniques, validation of security properties and their application to services and protocols.

Nina Yevtushenko joined Tomsk State University in 1991 as a professor and presently she leads a research team working on the synthesis and analysis of discrete event systems. Her research interests include formal methods, automata theory, distributed systems, protocol and software testing.

Mouni Lallali has got a PhD degree in 2009. He is assistant professor at University of Western Brittan (UBO, France) since 2010. His research interests are modeling and validation of Web/Cloud services, Multi-Tenant SaaS Architecture, and Multi-Tenant storage in the Cloud.