From Modeling to Test Case Generation in the Industrial Embedded System Domain

Aliya Hussain[1], Saurabh Tiwari[1], Jagadish Suryadevara[2], and Eduard Enoiu[1]

[1] Mälardalen University, Sweden
[2] Volvo Construction Equipment AB, Sweden
ahn16022@student.mdh.se, saurabh.tiwari@mdh.se,
jagadish.suryadevara@volvo.com, eduard.paul.enoiu@mdh.se

Abstract. Model-based testing (MBT) is the process of generating test cases from specification models representing system requirements and the desired functionality. The generated test cases are then executed on the system under test in an attempt to obtain a pass or fail verdict. While different MBT techniques have been developed, only a few target the real-world industrial embedded system domain and show evidence on its applicability. As a consequence, there is a serious need to investigate the use of MBT and the evidence on how modeling and test generation can improve the current way of manually creating test cases based on natural language requirements. In this paper, we describe an on-going investigation being carried out to improve the current testing processes by using the MBT approach within an industrial context. Our results suggest that activity and structure diagrams, developed under MBT, are useful for describing the test specification of an accelerator pedal control function. The use of MBT results in less number of test cases compared to manual testing performed by industrial engineers.

Keywords: MBT · Systems Engineering · Test Cases · Modeling

1 Introduction

Model-based testing (MBT) is an approach of automatically designing test cases based on behavioral models of system requirements [3]. These models represent the expected behaviour of the system under test (SUT). The testing process mainly consists of three high-level steps namely, *creation*, *execution* and *evaluation* of a test case. The test case *creation* is the most important part of this process as it involves the design of the preconditions, test steps and the expected output. The test case *creation* is a challenging activity and it has a direct impact on the ability to find faults and the quality of the resulting product. MBT automates the test creation by using abstract models developed at an earlier stage of the development process and promises to be a more efficient and effective method than manual testing [1][2].

In this study we present the results of an investigation at VCE (Volvo CE[3]), the stakeholder's requirements, needs, and concerns written in natural language.

[3] Volvo Construction Equipment AB, Sweden

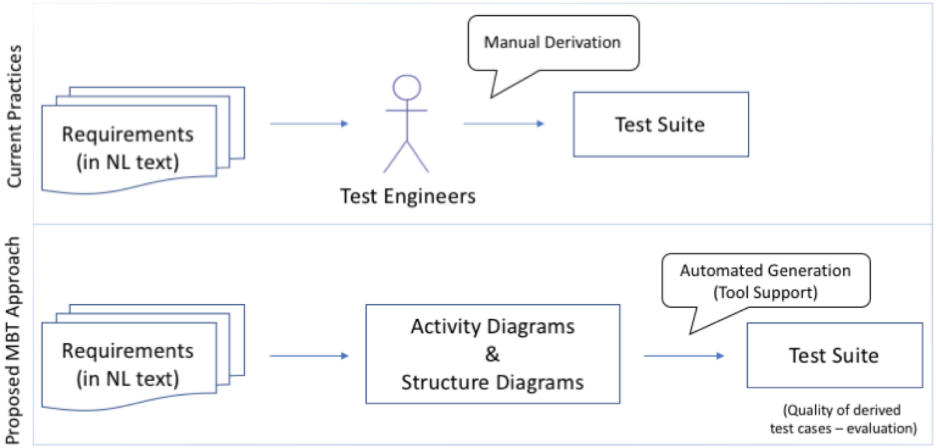

Fig. 1. Overview of MBT test case creation method and the current manual testing practice (NL stands for natural language).

Manual test cases are manually created using these requirements based on the test engineer's domain knowledge and experience. In Fig. 1, we illustrate the overview of how MBT differs in its high-level process to manual testing.

We describe the modelling and test generation process using Conformiq Creator as well as an exploratory comparison between manual testing and MBT in terms of number of test cases and test goal categories. The goal is to facilitate the use of automated test case creation using models of the system specification and show its applicability. We demonstrate how the MBT process is used (as described in Fig. 1) for modeling a realistic function controlling the *'Accelerator Pedal'* using activity diagrams (i.e, to specify the actions) and structure diagrams (i.e, to visualize the possible set of input and output parameters used). These diagrams are used by the MBT tool (Conformiq Creator[4]) to automatically generate test cases. Based on our initial investigations, we report our findings as well as point to future work.

2 Background

The study evaluates MBT use in an industrial scenario using a system provided by VCE. In this company, a management solution for systems engineering and software development (simply referred as the SE-Tool) is used as an adaption of the commercial tool *Systemweaver*[5]. This solution is a generic system modelling solution that supports the use of models (e.g., EAST-ADL[6] standard for automotive domain) and is a collaborative environment with support for system development. In this study we focus on the `Complete Analysis Function` (CAF) implemented in the SE-Tool framework and representing the functional

[4] https://www.conformiq.com/
[5] http://systemweaver.se/
[6] We refer the reader to the standard for further details: http://www.east-adl.info/

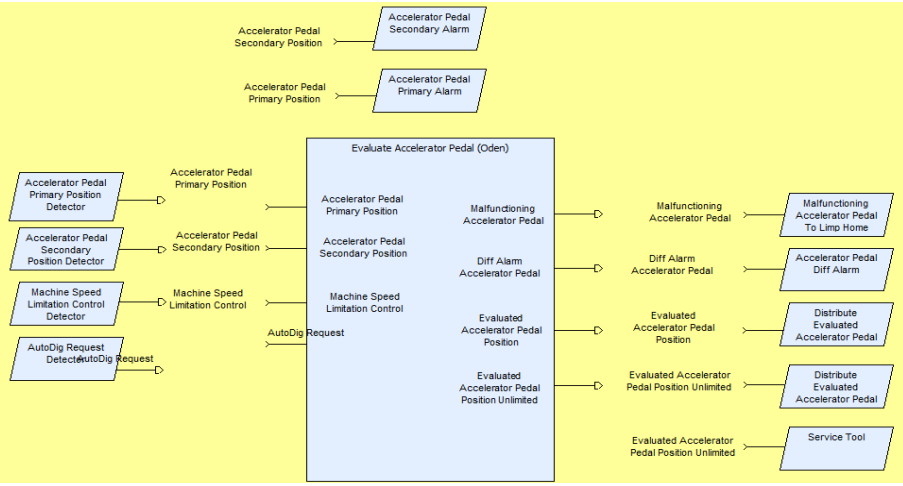

Fig. 2. View of Complete Analysis Function (CAF) for *Accelerator Pedal* function

architecture (i.e., the analysis level) of the Electrical and Electronic (E2E) control system w.r.to corresponding machine feature.

The CAF acts as a container for a collection of `Analysis Functions` (AF) and `Function Devices` (FD). An analysis function specifies a required function (within the E2E system) as a black-box mapping of inputs to outputs and a functional device (FD) that specifies the interface to other sub-systems, sensors or actuators. The HMI functional device is a special kind of functional device that is intended to be used for the operator interface; it defines components such as levers, switches and buttons for the operator interface. The SE-tool also provides the graphical overview of a CAF by showing all inputs and outputs as well as the interface with other subsystems.

The CAF function for the '*Accelerator Pedal*' is shown in Fig. 2. The purpose of this function is to evaluate pedal position requested from the *operator* (person who operates a machine through an appropriate interface). The inputs and outputs specified for the function are described, in terms of interface, as follows:

- ***Input Parameters:*** *Accelerator Pedal Primary Position*, *Accelerator Pedal Secondary Position*, *Machine Speed Limitation Control*, and *AutoDig Request*. The values of both *Accelerator Pedal Positions, primary and secondary,* are obtained form the two sensors attached to the "*Acceleration Pedal*". On the other hand, the *Machine Speed Limitation* function provides *Machine Speed Limitation Control values* and the *AutoDig function* provides the *AutoDigRequest values*.
- ***Output Parameters:*** *Accelerator Pedal Primary Position*, *Evaluated Accelerator Pedal Position Unlimited*, *Evaluated Accelerator Pedal Position*, and *Malfunctioning Accelerator Pedal value*. In a nutshell, the *"Accelerator Pedal"* function translates the accelerator request from the operator into the corresponding "propulsion force request" which is passed on to and finally actuated by the Drive-Line System (DLS).

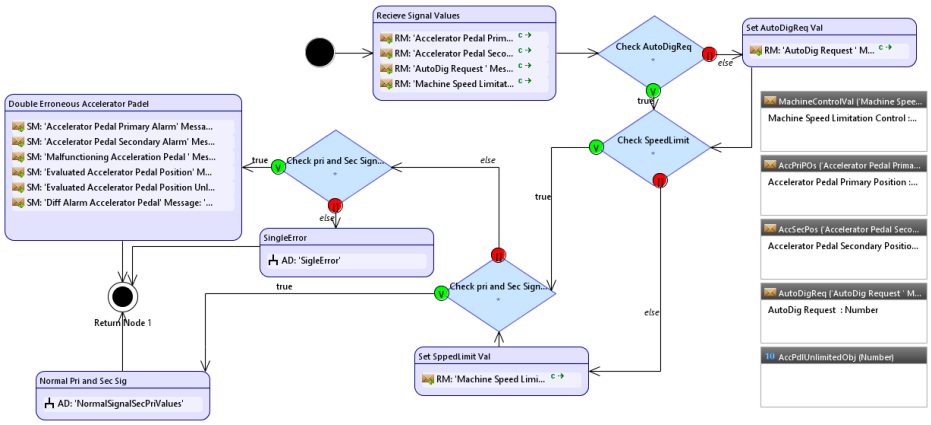

Fig. 3. An activity diagram showing the *Double Erroneous Accelerator Pedal* behavior.

3 The Modeling Approach

As shown in Fig. 1, in this paper, we describe a modeling approach to develop test models that enables automatic test case generation. The first step of the approach is to create, albeit manually (in future to be partly automated based on requirement models) a model from the CAF-based function specifications. Essentially, two types of models are created, namely, *activity* diagrams (i.e., representing behavioral models) and *structure* diagrams (i.e., representing a combinatorial model). We note here, that the structure diagram is limited to the input and output specifications. These diagrams created for the *Accelerator Pedal* function are described as follows:

- ***Activity diagrams***: The activity diagram shown in Fig. 3 specifies the system behavior corresponding to the "Double Erroneous Accelerator Pedal" state of *Accelerator Pedal*. In the first activity node all input values are initialized and saved in the corresponding data objects. The value of *AutoDigRequest* is checked if its value falls out of range. In case this is true, the variable is reset in the next state. Similarly, *Machine Speed Limitation Control* value is adjusted. In addition, the Accelerator Pedal Primary Position and Accelerator Pedal Secondary Position are checked. If both values are out of range the output is set accordingly.
- ***Structure Diagram:*** The structure diagram as shown in Fig. 4, is created based on the *Accelerator Pedal* CAF for defining the interfaces available for testing. Firstly, the inputs and outputs of the function are identified. The Accelerator Pedal Input Signal interface contains several message objects and each message object corresponds to a specific input of a function. The message objects in the Accelerator Pedal Output Signal interface specifies the function outputs.

In the next section, we describe the main results obtained using the modelled diagrams in the context of generating test cases as well as a comparison between MBT and manual test cases.

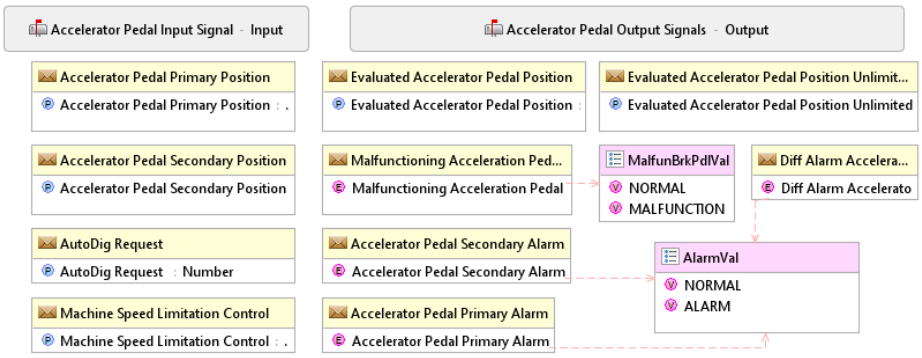

Fig. 4. Structure Diagram for *Accelerator Pedal*

Table 1. Example of the test case derived in the TIL2 format compatible with the VCE test environment

S.No.	Action	Expected Result
1	Accelerator Pedal Primary Position = 101; Accelerator Pedal Secondary Position = -1; AutoDig Request = -1; Machine Speed; Limitation Control = -1	Accelerator Pedal Primary Alarm = ALARM; Accelerator Pedal Secondary Alarm = ALARM; Malfunctioning Acceleration Pedal = MALFUNCTION; Evaluated Accelerator Pedal Position = Do not Care; Evaluated Accelerator Pedal Position Unlimited = -1; Diff Alarm Accelerator Pedal = NORMAL

4 Preliminary Results

The model of the Accelerator Pedal function is created manually and we validated the models by performing informal interviews with VCE test engineers responsible for testing the function under test to ensure model correctness and consistency. As a next step, the model is used as input to the Conformiq Creator tool to automatically generate test cases covering the created activity diagrams. Typically, the representation of a manually created test case in the SE-Tool is performed in the test instruction language version 2 (TIL-2)[7] format to facilitate the use of an automated test execution and evaluation environment. The test cases derived from the proposed MBT approach are exported to a compatible format (i.e., TIL-2). An example of a generated test case is shown in Table 1.

In order to contribute to the state-of-art, we compared MBT test cases with manual test cases created by industrial engineers. We conducted a preliminary empirical investigation in terms of covering different test goals and number of test cases. Together with several test engineers from VCE we defined six categories of test goals used when performing rigorous manual testing. These six categories are shown in Table 2 and cover a set of realistic testing goals for the function under test. Our results suggest that tests derived using the MBT approach are similar in nature and can be used to cover all test goal categories at a lower cost in terms of number of test cases created per each category; just for one of these categories (i.e., Erroneous Detectors (Single and double)) the number of

[7] ISO/IEC/IEEE 29119-3:2013; Software and systems engineering – Software testing – Part 3: Test documentation

Table 2. Test Goal Category and Number of test cases (TCs) comparison between manually created test cases by industrial engineers and MBT-based test cases.

Test Goal Category	**Manual TCs**	**MBT-based TCs**
Normal Operation	13	3
Differing Detectors	6	1
Pedal Position Output	8	2
Erroneous Detectors (Single and double)	5	3
Erroneous Autodig request and Machine limitation Control	22	3
All input erroneous combination	23	4
Total	77	8

test cases between the two techniques is similar. Overall, the total number of test cases created using MBT (i.e., 8 TCs) is significantly lower than for manual testing (i.e., 77 TCs). We have also found that these 8 TCs belong to multiple categories. A more detailed efficiency and effectiveness measurement would be needed to obtain more confidence in the results obtained in this study.

5 Conclusions and Future Work

In this paper, we present an investigation into the use of model-based testing in the embedded system context. We use the Conformiq Creator tool to model the behavior and structure of a function controlling the accelerator pedal provided by Volvo CE. We automatically create test cases covering the model and compare these test cases in terms of test goal coverage and number of test cases to assess the applicability of MBT in this context. The approach has shown encouraging results. As future work, we plan to also investigate the efficiency and effectiveness of MBT test-case generation. We plan to semi-automatically generate diagrams out of CAF specifications to reduce the effort of creating test models. In addition, we need to investigate the use of complex data types and timing aspects into the test model, since Conformiq Creator does not support decimal numbers or how to directly represent timing requirements.

Acknowledgments

This work is partially funded from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No. 737494 and The Swedish Innovation Agency, Vinnova (MegaM@Rt2). We would like to thank Kimmo Nupponen and the Conformiq team for their support.

References

1. Gudmundsson, V., Schulze, C., Ganesan, D., Lindvall, M., Wiegand, R.: An initial evaluation of model-based testing. In: 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). pp. 13–14 (Nov 2013)
2. Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., Stauner, T.: One evaluation of model-based testing and its automation. In: Proceedings of the 27th International Conference on Software Engineering. pp. 392–401. ICSE '05, ACM, New York, NY, USA (2005)
3. Schieferdecker, I.: Model-based testing. IEEE Software **29**(1), 14–18 (Jan 2012)