From Model to Rig – An Automotive Case Study

Josefine Södling[1], Rikard Ekbom[1], Peter Thorngren[2] and Håkan Burden[3]
[1]*Chalmers University of Technology, Gothenburg, Sweden*
[2]*Volvo Group Trucks Technology, Gothenburg, Sweden*
[3]*Viktoria Swedish ICT, Gothenburg, Sweden*

Keywords: Model-based Testing, Tool Adaptation, Organisational Change, Test Coverage.

Abstract: As the size and complexity of the software in a truck grows, new ways of managing the development are needed. Numerous reports show how MDE can be successfully applied for automotive software development. We complement previous research by conducting a case study on the impact of model-based testing for verifying and validating the behaviour of a truck's headlights. Our results are three-fold. First, we show how a model can be transferred from a model-in-the-loop setting to a hardware-in-the-loop via system simulation. Second, we supply an analysis of the shortcomings of the model that were found as the model was tested in more and more platform-specific settings. Third, our results show that the introduction of model-based testing practices will require organisational changes even if the used tools are familiar to the company.

1 INTRODUCTION

The automotive industry is currently in a shift from being a hardware-centric industry to becoming a software-intense domain (Bringmann and Kramer, 2008) – soon 90% of automotive functions are developed as software. Subsequently, the integration, validation and verification of the code becomes more and more complex. But it also means that the latency between specification and testing of new features can be drastically shortened when functionality, in terms of software, is decoupled from the hardware development.

Model-based testing, MBT, is one way of automating and scaling software testing in the automotive industry (Han et al., 2013). MBT starts in the Model-in-the-Loop stage, MiL, where a model with the sought behaviour is defined. A benefit of starting the testing at this stage is that defects and inconsistencies require less time and effort to be analysed and fixed than in a full-fledged mechatronic system (Pretschner et al., 2005). The next stage is referred to as Software-in-the-Loop, SiL, and here the surrounding system is simulated to validate that the model behaves as expected in its context (Schieferdecker, 2012). When the tests confirm that the model behaves as expected the model is transferred into the Hardware-in-the-Loop stage, HiL. Here the model's behaviour is validated in relation to a hardware rig. Including proper mechatronic systems and hardware into the test environment means that the cost of testing increases, but so does also the probability that the model will behave as expected on the designated platform. The different stages and their relations are shown in Figure 1.

A recommended way of introducing model-based practices such as MBT is to start with a small and well-known subsystem whose behaviour can be fully monitored through the test stages instead of using a large and complex model that cannot progress fully through the test environments (Schieferdecker, 2012). When the smaller subsystem is verified and validated, the model can be complemented with additional features and constructions. Introducing MBT can be a costly affair considering the amount of licenses, hardware parts and new competencies that the company has to invest in (Whittle et al., 2013). This can be mitigated by reusing existing tools and building on established skills within the company (Utting and Legeard, 2010).

Through an exploratory case study (Runeson et al., 2012) we set out to define a a set of practices for iteratively refining a model to a more and more platform-specific context at Volvo Group Trucks Technology. In this way a broader set of details and software concepts can be validated and verified. Based on the findings of our study we set out to answer the following three research questions:

Södling, J., Ekbom, R., Thorngren, P. and Burden, H.
From Model to Rig – An Automotive Case Study.
DOI: 10.5220/0005644506150622
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 615-622
ISBN: 978-989-758-168-7

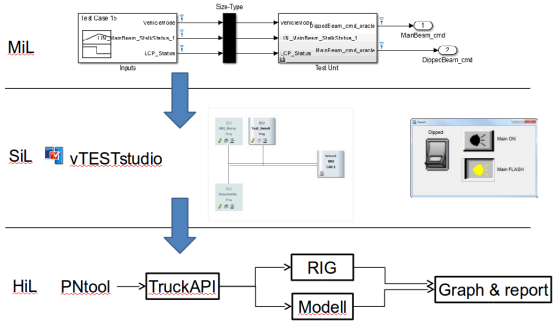

Figure 1: From MiL to HiL.

RQ1: Which shortcomings of the model are exposed as the testing is conducted in a simulated system and in a hardware rig and how can these shortcomings be mitigated?

RQ2: How will the changing test environment impact the test coverage?

RQ3: How can the resulting test practices be implemented in the organisation, making sure that testing is both trustworthy and reusable?

Our results are three-fold. First, we show how a model can be transferred from a Model-in-the-Loop setting to a Hardware-in-the-Loop context via a system simulation. Our findings include how a model can be reused as the central test object from the modelling environment to a rig of hardware via system simulation. Second, we supply an analysis of the test coverage as the model was tested in more and more platform-specific settings. Third, our results imply that the introduction of model-based testing will require organisational changes even if the used tools are familiar to the company.

2 RELATED WORK

Two independent studies (Kuhn et al., 2012; Aranda et al., 2012) report on applying Model-Driven Engineering, MDE (Kent, 2002), at General Motors. The former publication focuses on the individual perceptions of the adoption of MDE at General Motors; the latter reports on how MDE induced changes at the organisational level. At the individual level, engineers experienced both forces and frictions related to MDE tools and languages – for example, a lack of support for developing the MDE infrastructure. At the organizational level, Aranda et al. found, for instance, that software developers were now asked to focus on MDE infrastructure whereas the domain experts became the ones who implemented the new functionality.

In a comparison of MDE at three large companies, Burden et al. conclude that MDE can empower the domain experts to be the primary software developers (Burden et al., 2014). A pre-requisite in the case of the automotive industry is that the engineers are trained in using the relevant modelling tools during their university education and that there are other engineers who can develop and maintain the infrastructure needed to transform the high-level architecture into the modelling environment and from there on to the designated target.

In relation to our own study a similar case has been investigated at Volvo Cars Cooperation. While the two companies share the same name they are distinctively separate companies with their own organisations, products and services. In the Volvo Car case MBT was seen as an enabler to shorten the development cycles in combination with an agile way of working (Eliasson et al., 2014). Furthermore, they conclude that by developing new functionality in a virtual test environment the dependencies on hardware and external suppliers can be postponed until a later occasion when the external deliveries are more trustworthy.

3 METHOD

We conducted an exploratory case study (Runeson et al., 2012) regarding the main beam functionality to better understand how MBT can be implemented at Volvo Group Trucks Technology.

3.1 Context

The case study was conducted at Volvo Group Truck Technology's facilities in Gothenburg, Sweden. Today most of the software development is outsourced while in-house development is spread around the world, including India and France. The distributed development has an impact on the overall integration carried out in Gothenburg since bugs that are found late are expensive and time consuming to fix.

The incentive behind our case study is that Volvo wants to manage the complexity of truck development by finding defects and inconsistencies earlier than today. The existing process at Volvo can be described as a waterfall with a sequential progression through analysis, design, implementation, test and integration before delivery. In the current flow, errors are often not identified until integration which in turns means that the corrective actions become expensive and have a negative impact on the overall progression. In parallel, an average truck consists of 70 or more Electronic

Control Units, ECU, that are tailored for specific purposes. One way of handling the growing complexity of the truck is by using fewer, general-purpose ECUs. This will in turn demand a new way of developing the features of the truck where the testing is just as generalised as the new architecture. Here MDE has been identified as the way forward and subsequently model-based testing needs to be evaluated before it is launched full-scale within the organisation.

Testing is not just an activity for validating and verifying new functionality, testing is carried out on both new and old truck models as features are improved over time. This means that the test method has to be adapted for both development of new features and for maintenance of old models where the latter often lack some of the software found in modern trucks.

3.2 Model-in-the-Loop

The model that was used for the case study was developed using Simulink. Since the Volvo Group also owns truck brands such as Terex Trucks, Renault, Dongfeng and Mack, besides Volvo, it was important that the model was platform-independent (Mellor et al., 2004) in order to be reused across the brands and platforms. The behaviour of the model was defined through a set of requirements already present at Volvo. The implementation was based on boolean logic that specified the possible combinations of the incoming and outgoing signals. Due to the fact that the model did not include platform-specific details and the signals only had a fixed number of values it was possible to test all possible combinations of incoming and outgoing signal values for the model.

The testing of the model was conducted by applying an additional package to Simulink, the Design Verifier tool. Design Verifier helps to find anomalies in the model and generates complementary test cases for the anomalies. Here examples of anomalies would be dead logic or division by zero. We also used Simulink's package for code generation to generate Dynamic-link Library-files, .dll-files.

3.3 Software-in-the-Loop

CAN Open Environment, CANoe, is a tool developed by Vector which is used for development, testing, simulation and analysis of software for the automotive industry. The tool can be used to simulate a network of ECUs connected through a Controller Area Network (CAN (Etschberger, 2001)). When a physical network is used the tool also allows for sending a signal across CAN at the same time as monitoring the network to record and assess the impact of the communication. In order to integrate CANoe with Simulink, the tool vendor Vector has developed their own Simulink blocks that allow the inclusion of CANoe concepts into the Simulink model. We also used vTESTstudio to generate tests that cover all possible combinations of the incoming signal values.

By using the designated CAN-blocks in CANoe we could configure a virtual CAN for the Simulink model. The configuration only included the most relevant ECUs for testing the incoming and outgoing signals of the model. The network was displayed in a graphical interface in order to monitor the changes during each test case. To prevent hardcoding the configuration to fit the model we defined our own system signals in CANoe that can easily be reused with a new configuration. The system signals where then connected to the corresponding incoming and outgoing signals of the model. This enabled us to stimulate the model with CAN signals, read the results from the model and validate that the behaviour was consistent with the requirements. We opted to use vTESTstudio to define the test cases, again. After each test case has been executed the outcome from the simulated environment is logged together with the result given by the model. The two values are then compared and when all test cases have been run CANoe generates a report, declaring the success or failure for each test case.

3.4 Hardware-in-the-Loop

The rig used for the HiL stage consisted of relevant parts from a Volvo truck (physical controls such as dials, levers and buttons as well as displays) which in turn were connected to a rig of ECUs, computers and real-time simulators which in turn relayed the control commands to two head lights mounted on a board. In this way it is possible to visually monitor the outcome of different test cases. The communication in the rig is implemented as a physical CAN, just as it would be in a real truck. The inclusion of the computers enable the testing to be monitored and controlled. For instance, signal values can be set from the computer or the state of the ECUs can be monitored without disturbing or affecting the communication on the CAN.

Besides the tools from Vector and Mathworks a number of tools developed in-house were used. An example of such a tool is PNTool that together with the Truck Control API enable interaction with the rig through the computers instead of using the dials and buttons. In order to port the model to the rig it is necessary to have a C compiler that is compatible with the used MatLab version. The model is then synthesised

(Mens and Gorp, 2006) into a dll-file targeted for the CANoe platform. The generated file is then incorporated with a simulated ECU and integrated into the CANoe-environment. Then the CANoe configuration is updated to accommodate both the simulated system and the system signals of the physical rig. This time we used PNTool for specifying the test cases. A drawback with PNTool compared to vTESTstudio is that the first lacks the possibility to compare values of system signals. It is therefore not possible to automatically compare the values generated by the model with the values read from the physical rig. Instead a visual inspection was done for each test case.

3.5 Evaluation

In order to evaluate the implementation of MBT continuous meetings and discussions were held with a reference group. The group consisted of one of the engineers responsible for developing and maintaining the rig, two engineers responsible for testing and three employees from the division responsible for integration. The evaluation covered the current situation at Volvo in terms of the engineers competence in testing, the availability of tools and licenses as well as which components where already in place for the hardware rig.

4 RESULTS

The results are structured in accordance to our research questions so that the first subsection answers *Which shortcomings of the model are exposed as the testing is conducted in a simulated system and in a hardware rig and how can these shortcomings be mitigated?*; the second subsection highlights the contributions in response to *How will the changing test environment impact the test coverage?*; while the question *How can the resulting test practices be implemented in the organisation, making sure that testing is both trustworthy and reusable?* is answered in the third subsection.

4.1 From MiL to HiL

The results regarding the transition from model to rig are broken down in terms of model to simulated system, and then simulated system to hardware rig.

In order to fit the model into the simulated system environment we had to replace the existing Simulink ports with CANoe blocks, otherwise we could not monitor the communication on the simulated network. When substituting the incoming ports we found the first inconsistency, a mismatch regarding datatypes. The Simulink model assumed that the incoming system variables should be of the type `integer` while they in fact are of the type `double`. This was easy to fix by using the converter block supplied by Simulink.

The test cases defined at the MiL stage could not be reused and we had to manually redefine all test cases using vTESTstudio. The reason for reimplementing the tests was that there was no possibility to map the model elements to CAN representations using Simulink Design Verifier. After executing a test case we checked that the model and the simulated system generated the same result. Since all test cases returned the same output for both model and simulated system we could continue to the next stage, Hardware-in-the-Loop.

The first thing that happened when transitioning the model to the hardware rig was a conflict regarding CANoe versions. This meant that we could not reuse the test cases from the earlier stage and again had to manually redefine them. This time we used PNTool together with the Truck Control API for implementing the test cases. In this way we could use the signals provided by the API to simulate the physical stimulation, such as turning a dial or pressing a button.

The CANoe configuration used for the simulated system could be adapted to the hardware rig since it was not hardcoded for a specific setting. In the new configuration the model resided on a simulated ECU. We then ran our test scripts and since PNTool lacks the ability to compare system variables the resulting graphs were manually compared and analysed. During the analysis of the graphs it became evident that the model had some serious shortcomings. In this case the reason was a combination of human factors, such as not understanding or taking all requirements into consideration during the development of the model, and lack of information regarding the dependencies in relation to surrounding systems.

One of the most important deviancies originated from assumptions regarding the logic behind the Exterior Light Control dial, which has an internal variable to keep track of its current state. The model assumed that the dial would submit the current state when it in fact only signals how many steps it has moved (anti-)clockwise.

Another finding was that the Main Beam stalk behaved differently than the model assumed. The model requires a constant feed of signal values in order to determine what the outgoing values should be. When the stalk is retracted to its end position it generates a signal value representing head beams on. When the stalk is released it goes back to its neutral position but

the lights should still be on. This was how the lights in the hardware rig behaved. The outgoing values of the model on the other hand said that the lights were off. This was due to the signal values changing when the stalk went into neutral mode.

In order to mitigate these shortcomings of the model it had to be complemented by code that kept track of the ELCP's current state and translated turning the dial into a new state. It also meant changing the model to accommodate the logic of the stalk.

During the evaluation it was recognised that the logical shortcomings of the model were much easier to expose while testing on the hardware rig than in the earlier test stages.

4.2 Test Coverage

For MBT to be profitable the model has to be trustworthy beyond a certain degree. One way of achieving this is to scope the model to a specific subsystem, such as the main beams. As mentioned in section 3.2, the original test cases where developed using Simulink Design Verifier which delivered full test coverage for all the possible combinations regarding the values of the main beams. It also filtered out test cases which do not add any extra coverage in order to save time during batch testing. For the model to be reliable throughout the testing and all configurations it is important that the right test cases are carried through the transitions as well. The model could be transferred between stages since the involved tools provided the necessary transformations for each target test environment. As we saw in section 4.1 his was not true for the test cases.

While it was possible to redefine all the test cases using vTESTstudio for the simulated system at the SiL stage, it was not possible to automatically filter out superfluous test cases. In our case this had a limited impact on the time needed to run the test suite since the number of combinations to test was limited. The test coverage was unchanged.

If porting the test cases from MiL to SiL was trivial, transferring the test cases to the hardware rig was more challenging since neither Simulink nor CANoe has relevant support. In Simulink the test cases are defined inside a shared block. After each test case is executed the internal variables are overwritten without being stored over time. In the end the solution was to write a script that saved the values in a table using Microsoft Excel. Just as for the SiL stage, it was possible to transfer the test cases and achieve full test coverage. Even if it required more effort than desired it was therefore possible to retain full test coverage from the MiL to the HiL stage.

4.3 Organisational Impact

Currently, an engineer needs to know and handle a number of different tools, know the product and be a competent tester in order for MBT to be successful. At Volvo the engineer also needs an understanding of how to operate the hardware rig. If MBT is to be successful at Volvo it is necessary that only a few engineers who have all the desired skills work with the development and maintenance of the test environment since so many unique competencies are required. Then the designated testers do not need the full spectra of skills but can instead focus on developing, executing and analysing the test cases.

For MBT to be successful throughout the organisation, Volvo needs to invest in training the engineers in the new way of working, acquiring the necessary tools and licenses that are not present at now but also a continuous integration of new features into the test environment as the product to be tested evolves. For this to happen it is vital to formulate a plan on how the development is to be managed and and that this information is successfully spread throughout the organisation.

Our aim was to automate the testing procedure in order for it to be reusable. In our case reuse is both in terms of applying it to the model after it has been changed as well as for the practices being applicable to other models. In the case of reusing the method on a model after it has been changed, it is necessary that the engineer has knowledge of the modelling and testing tools used at the MiL stage. But the engineer also needs sufficient knowledge about the requirements regarding the new model details. In this setting human factors is a risk since the models are not automatically generated from the requirements. And a model including many inconsistencies will require more time and effort to get accepted at the SiL and HiL stages.

In order to reuse the test practices for a new model it is necessary to create a new configuration to integrate the model with the system signals. For the SiL stage this is straightforward while the HiL stage requires knowledge of how the signals are used in a physical truck. The configuration gets more complex as the number of system signals and ECUs grows.

5 DISCUSSION

While the objective of the case study was achieved the process of getting there could have been more straight-forward. One area of concern is the poor interoperability of the applied tools. For instance, transferring the test cases from one test stage to another

which had to be done manually. This shows that even if it requires time and effort, it is possible to create a Simulink model that is comparable to the test results of a hardware rig. Thus, the testing at Volvo can be generalised by model-based testing. Our results also indicate that the SiL stage had lesser impact on identifying defects and inconsistencies than testing at the HiL stage. If the same is true for a more complicated model is still to be explored. From the point of cost it would be desirable that the SiL stage became more influential since it is less expensive to develop and maintain and the feedback loop to the MiL stage is shorter. While it was at the HiL stage the most severe shortcomings of the model were identified, it is worth remembering that each stage contributed to exposing errors in the model.

After the test results have been evaluated it is possible to go back to the MiL stage and add new models, requirements or features to expand the scope of the test. At each step knowledge of the intended behavior of the head beams feature and the truck domain in general was needed to assess the test results. If the model returns a different verdict than that from the simulated system or the physical rig it is not given what is right or wrong. This means that in order to define relevant corrective actions or to stay with the model as it stands requires another set of skills and competencies than those used for developing and maintaining the test environment.

In our setup the model has been used as a comparator for evaluating the test results from the simulated system and the hardware rig with those of the model itself. Another approach would have been to use the model as the control logic at each stage, so that the model would determine the behaviour of the head beams at each stage. In such a setup the test method would evaluate to which extent the model represented the sought behaviour of the truck.

Whittle et al. argue that while there are plenty of modelling tools around, few of them are mature enough to be used without costly adaptations (Whittle et al., 2013). The availability of tools and competencies within the organisation lowers the risk and cost of introducing a model-based way of working (Burden et al., 2014), but has to be balanced against the possibility that the tools will not be sufficient for the new purposes (Whittle et al., 2013). In our case we found that there was a lack of tools that fitted our purposes which required tools to be developed in-house or the help of external tool vendors to fit the existing tools to new practices. A substantial part of the work has been conducted using CANoe and subsequently the development of the test cases and test environment was done in close collaboration with representatives from Vector, the tool vendor. On the positive side of this collaboration is that Vector could develop new functionality and tailor CANoe to our needs as new insights were gained regarding the test method. It also meant that Vector could be close at hand for additional training in how to best apply their tools and use their add-ons. On the negative side there is often a delay when interacting with external developers. This is due both to the manual handover of the problems but also to the fact that determining which tool vendor to address takes time in itself. For instance, if the transformation of test cases from one tool to the other does not work as wished, should we ask for the source or target tool to adapt our requests? Besides, if all the involved tools had been developed by the same tool vendor they would hopefully allow for smooth integration of features and automation of tasks. A subset of the used tools, tool vendors and their inter-dependencies are shown in Figure 2.

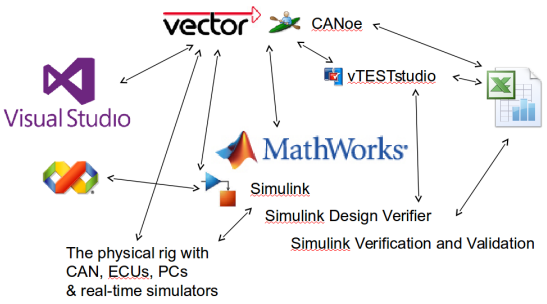

Figure 2: A subset of the used tools and technologies with their major dependencies.

In relation to previous studies we find a lot of commonalities. Just as Aranda et al. found at General Motors (Aranda et al., 2012), there is an organisational impact from introducing model-based technologies and ways of working. In our case the test environment will have to be developed and maintained by different engineers than those that do the actual testing. The root cause is that there are not many engineers that have the necessary skills and competencies to both develop a test infrastructure as well as model the beahiovur of a truck – the skill sets are too orthogonal. And as we have seen, the domain knowledge is instrumental in getting the model right from the beginning, which supports earlier claims by Burden et al. (Burden et al., 2014) and Utting and Legeard (Utting and Legeard, 2010).

As previously shown (Burden et al., 2014; Utting and Legeard, 2010), the development of the supportive infrastructure will have to be synchronised with the needs of the test engineers as new features and configurations will place new demands on the test environments. If this will lead to organisational and social tensions within the company, as in the case at

General Motors, is still an open question.

Throughout the process it also became evident that when the model is done up-front, before a full understanding of the surrounding system is in place, it will be based on assumptions on what the context will look like. This is in line with results from Volvo Cars (Eliasson et al., 2014). In our case we found inconsistent assumptions regarding the types of signal values as well as the internal logic of surrounding systems.

Finally, introducing MBT will have a positive effect on the time it takes to go from concept to production since testing can be done much earlier and the combination of simulations and hardware rigs will expose defects quicker than the traditional waterfall process. In this way model-based ways of working will promote an agile and iterative development, which in turn shortens the lead times (Eliasson and Burden, 2013; Burden et al., 2014).

6 CONCLUSIONS

Our study complements and expands previous research by exploring the impact of model-based testing for validating the behaviour of a truck's head beams. The study was conducted at Volvo Group Trucks Technology in Gothenburg, Sweden, and consisted of testing at three different stages. First the model was tested as is at the Model-in-the-Loop stage; then it was tested in a simulated system known as the Software-in-the-Loop stage; before finally being tested in a hardware rig – the Hardware-in-the-Loop stage.

Our results are three-fold. First, we show how a model can be transferred from a Model-in-the-Loop setting to a Hardware-in-the-Loop context via a system simulation; the overall process is shown in Fig. 1. Second, we supply an analysis of the shortcomings of the model that were found as the model was tested in more and more platform-specific settings. Third, our results show that the introduction of model-based testing will require organisational changes even if the used tools are familiar to the company.

In the near future we will be able to test the model on a real truck to see if any new defects or faulty assumptions are exposed. Another research direction for the future is to explore to which extent the test cases developed for the MiL stage can be automatically reused at the more platform-specific stages, but also to determine to which extent the test cases for the SiL and HiL stages can be reused with other programming tools since it is not yet established which tools will be used across the organisation.

Speaking of organisation, it is still work in progress to implement and adapt the developed test practices in the organisation at large and we aim to report on the organisational changes and necessary adaptations of model-based practices. A key aspect to explore is to which extent MBT can be used for other features of the truck and in other divisions of the organisation. In this line of work we will seek strategies to mitigate the tensions reported on by previous studies, to explore how the test engineers and test environment developers can work in harmony with each other.

ACKNOWLEDGEMENTS

The authors would like to thank Mathworks for providing the necessary licenses for free during the case study. We would also like to acknowledge the effort that Vector put into the case study by adapting their tools for our needs. This work was partially funded by the Vinnova project Next Generation Electrical Architecture.

REFERENCES

Aranda, J., Damian, D., and Borici, A. (2012). Transition to Model-Driven Engineering - What Is Revolutionary, What Remains the Same? In *MODELS 2012, 15th International Conference on Model Driven Engineering Languages and Systems*, pages 692–708. Springer.

Bringmann, E. and Kramer, A. (2008). Model-Based Testing of Automotive Systems. In *Software Testing, Verification, and Validation, 2008 1st International Conference on*, pages 485–493.

Burden, H., Heldal, R., and Whittle, J. (2014). Comparing and Contrasting Model-driven Engineering at Three Large Companies. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, pages 14:1–14:10, New York, NY, USA. ACM.

Eliasson, U. and Burden, H. (2013). Extending Agile Practices in Automotive MDE. In *XM Extreme Modeling Workshop*, Miami, Fl, USA.

Eliasson, U., Heldal, R., Lantz, J., and Berger, C. (2014). Agile Model-Driven Engineering in Mechatronic Systems - An Industrial Case Study. In Dingel, J., Schulte, W., Ramos, I., Abrahão, S., and Insfrán, E., editors, *Model-Driven Engineering Languages and Systems - 17th International Conference, MODELS 2014, Valencia, Spain, September 28 - October 3, 2014. Proceedings*, volume 8767 of *Lecture Notes in Computer Science*, pages 433–449. Springer.

Etschberger, K. (2001). *Controller Area Network: Basics, Protocols, Chips and Applications*. IXXAT Automation GmbH.

Han, K., Son, I., and Cho, J. (2013). A study on test automation of IVN of intelligent vehicle using model-based testing. In *Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on*, pages 123–128.

Kent, S. (2002). Model Driven Engineering. In *Proceedings of the Third International Conference on Integrated Formal Methods*, IFM '02, pages 286–298, London, UK. Springer-Verlag.

Kuhn, A., Murphy, G. C., and Thompson, C. A. (2012). An exploratory study of forces and frictions affecting large-scale model-driven development. In *Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems*, MODELS'12, pages 352–367, Berlin, Heidelberg. Springer-Verlag.

Mellor, S. J., Kendall, S., Uhl, A., and Weise, D. (2004). *MDA Distilled*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Mens, T. and Gorp, P. V. (2006). A Taxonomy of Model Transformation. *Electronic Notes in Theoretical Computer Science*, 152:125–142.

Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., and Stauner, T. (2005). One Evaluation of Model-based Testing and Its Automation. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 392–401, New York, NY, USA. ACM.

Runeson, P., Höst, M., Rainer, A., and Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons.

Schieferdecker, I. (2012). Model-Based Testing. *IEEE Software*, 29(1):14–18.

Utting, M. and Legeard, B. (2010). *Practical Model-Based Testing: A Tools Approach*. Morgan Kaufmann.

Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., and Heldal, R. (2013). Industrial Adoption of Model-Driven Engineering: Are the Tools Really the Problem? In *MODELS 2013, 16th International Conference on Model Driven Engineering Languages and Systems*, Miami, USA.