Improving Dependability of Embedded Software Systems using Fault Bypass Modeling (FBM)

Book Title will be set
Book Editors will be set
PTI, 2015

Improving Dependability of Embedded Software Systems using Fault Bypass Modeling (FBM)

Rakesh Rana, Miroslaw Staron, Christian Berger
Department of Computer Science and Engineering
Chalmers | University of Gothenburg
Gothenburg, Sweden
rakesh.rana@gu.se

Abstract. Fault injection techniques are important and widely used for verifying the dependability of computer systems. Traditionally fault injection has been successfully applied for evaluating dependability of hardware electronics and is now increasingly been used for software systems. At the same time increasing complexity of embedded software systems such as in automotive sector has driven these domains to use Model Based Development and using virtual simulation to build and test models before actual code is generation from these models. In this paper we conclude that fault injection techniques can be effectively used for assessing and thus increasing the dependability of embedded software systems and analyze a problem that is faced when using fault injection within a virtual simulation of these systems. We also discuss a framework referred to as Fault Bypass Modeling (FBM) as one possible solution to the described problem with the help of autonomous vehicle simulation case study.

Keywords. Fault injection, testing

1. Introduction

Embedded software today plays a very significant role in our daily lives. Everything from our mobile devices, telecommunications infrastructure, satellites to home appliance and automotive products depends heavily on the embedded software to provide their functionality and services. Over the last two decades, there has been an enormous increase in the complexity of embedded software, shorter innovation cycle times, while the demands for their reliability and dependability have anything but grown [1].

Due to requirements of real time behavior and stringent demands for quality and dependability, embedded software's are much more complex than their counterparts in IT applications or desktop software. Also given that the late defect correction costs higher in embedded software development and testing of software after code completion costs about 30-50% of all resources [2], verification and validation holds special significance in this domain. Model driven or model based development (MBD) is now widely adopted within the domains of embedded software/systems development. Good overview on embedded software development and model based approach within it can be found in [3], [4], [5], while [2] provides important facts, figures and the expected future for embedded software.

The problem and challenges related to verification and validation of models, specifically how to verify, validate and test the behavioral/implementation models that are used for code generation is also well recognized within the research community of model driven engineering [6]. The predominant form of testing within embedded software using MBD is done using test cases and test scenarios. Model based testing (MBT) approach attempts to use data models to generate the tests where data models intends to capture the requirements and input configurations [7]. While test automation and MBT provides considerable reduction in cost of test generation, importance of real-time issue and need for testing using continuous signals calls for reactive or closed loop testing.

Closed loop testing offers many advantages for testing systems which depend or interact closely with their environment. By modeling the environment and interacting the system under test with its environment through controlled interface(s) - provides possibility of reactive testing, identification/generation of multiple system-environment test cases/scenarios automatically and tests the system for its real time characteristics.

Fault injection techniques can further enhance the effectiveness of closed loop testing and thus help in evaluating and increasing the dependability of system in their early stage of development, but injecting faults into system (in closed loop configuration) may make system behavior and its output unrealistic and thus unreliable for making analysis or testing hypothesis. The problem occurs mainly due to dependencies between the system and its environment and feedback loops between the two. In this paper we highlight the problem and discuss how framework referred as fault bypass modeling can be used as a potential solution to this problem.

2. Related work

The fault bypass modeling idea presented here is introduced by the authors of this paper in [8], with a case study using a behavioral model of anti-lock braking system in Simulink. In this paper we highlight the need of closed loop testing and evaluate the applicability of fault bypass principle to a case of virtual simulation of autonomous vehicle case

Although using fault injection techniques for dependability evaluation of behavioral/functional models is at its infancy, Svenningsson et al. [9] introduce the tool called MODIFI which can be used to apply fault injection methodology on functional/behavioral models in Simulink. The tool is capable of injecting single or multiple faults into the signals of a given system to evaluate the fault propagation properties and analyze the effectiveness of fault tolerance mechanism of system under test. But as described in our earlier work [8], using such tools in closed loop mode need careful consideration to fault bypass principle to ensure that system output is realistic and reliable. Such a modelling is important even for other domains, e.g. measurement system or software modelling in general [10, 11].

Trawczynski et al. [12] presented an approach for modelling software systems in cars using closed loops in the context of security engineering. Their approach provides another example from a similar domain.

2.1. Need for closed loop testing

The main form of traditional software testing is open loop testing using test case and scenarios approach. But in number of industrial domains and types of applications where

program/system under test is non-deterministic or where the behavior of given function/system depends on its operational environment – open loop testing is not an effective approach. In such cases the problem of test case generation even using the MBT or automated test generation tools is much more complex than for deterministic type of applications [13].

Stockmann et al. [14] documents the need for closed loop testing in the automotive industry. Focusing on the domain of electric vehicles and testing electronic control units (ECUs), the authors propose methodology and tool chain for simulating virtual ECUs to enable functional testing under different conditions. Requirement of using closed loop testing for testing model based development in automotive domain due to real time issues behavior and need for using continuously changing signals is also expressed in [15]. The need for testing in the virtual space (in closed loop configuration) due to advancements in autonomous driving, vehicle to infrastructure and vehicle to vehicle communication is further established in [16].

The problem of non-deterministic factors of testing and need for closed loop testing in the area of medical devices is explained and highlighted in [13]. The authors refer implantable devices increasing complexity as a factor leading to large amount of device recalls. Safety critical nature of such systems calls for effective testing, to allow for physically relevant model based test generation for such devices they describe a closed loop testing environment for testing pacemakers. The capability and effectiveness of the approach is demonstrated by system's ability to test for common and complex heart conditions for different pacemaker models.

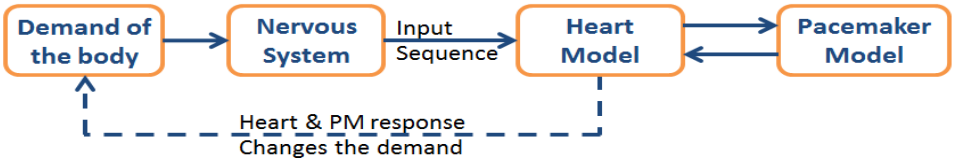

Fig 1: Simulated heart and pacemaker model in closed loop configuration as presented in [13].

A common approach to test systems with human elements in closed loop is to couple the human subject to the simulated system. Using virtual human models as a cheaper alternative is an area of active research [17]. Thus closed loop testing is also important in areas with man-machine interface which constitutes a large part of day to day products with embedded software.

2.2. Fault Injection

Fault injection has been used with good results for verification of dependability attributes of hardware and software systems [18]. Fault injection is widely used to identify bottlenecks related to dependability, study the behavior of system under faulty operating conditions and examining the coverage of fault tolerance or error detection and recovery mechanisms within software systems.

For applications which are safety, mission or business critical, dependability evaluation is especially important activity. Fault injection techniques have been much studied and used for safety critical applications development. SCADE or Safety-Critical Application Development Environment is a modeling language developed to simulate hardware failure scenarios. SCADE have been used in projects such as ESACS and

ISAAC for identification of fault combinations leading to safety case violations. A plug-in called FISCADE [19] have also been developed for SCADE language for introducing faults into using the SCADE simulator.

3. Improving closed loop testing using fault injection

As described in 2.1, there is high need for using closed loop testing for number of domains and applications. Closed loop testing can be achieved by developing/modeling the environment (with which the system interacts) and simulating the system and environment coupled through interfaces in the virtual space. Using MBD and MBT approaches in conjunction can be used to generate tests for system in closed loop configuration which works well under normal (specified) working conditions.

But in order to achieve dependability evaluation of a system; for example running a fault based scenarios, we need to go one step further to the closed loop testing. This could be easily done by injecting faults into the system many scenarios can be created for instance a system with inputs from n sensors may run scenarios with individual failure of x $(0 < x < n)$ sensors input and their combinations. Different types of sensor/input failure modes could be modeled and so does the failure related to reading parameters and system dependencies onto other system which simulates reading, writing or memory errors. All these fault operating conditions can be used to identify failure modes under which system output is unacceptable and test cases/scenarios generated to ensure that final implementation code have error handling or tolerance capabilities to avoid such scenarios.

Thus by coupling fault injections techniques with close loop testing, the efficiency and effectiveness of testing real-time systems with non-deterministic or environmental dependent properties can be enhanced significantly. Model based development and closed loop configuration allows for automated running the system against large number of normal and fault scenarios which are not possible in open loop configuration or using manually crafted test cases.

But the main challenge in using fault injection in a close loop configuration is to differentiate between correct system's behaviors from the system failure under fault mode. The problem is described in the next section using a simple case study, while detailed description using a behavioral Simulink model is also available in [20].

4. Case study: problem description and proposed solution

In this section we describe the challenge when using fault injection in close loop configuration. We use the miniature vehicle/car and its environment model described in [16].

The implementation of system-environment model for the autonomous miniature car and its environment is represented in Fig 2. The modules named monitor, lanedetector and driver are the parts constituting the system within the car, while the vehicle, camgen and irus forms the environment simulator (environment model). The environment simulator can take inputs form scenario modeling GUI which gives flexibility of designing and running test scenarios.

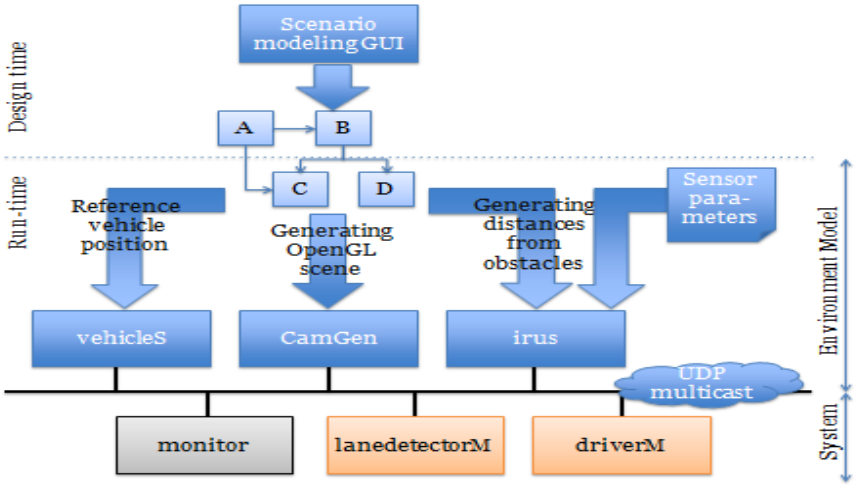

Fig 2: Representation of model-based system-environment model capable of simulating vehicle-environment model in virtual space, as presented in [16].

When simulating the autonomous miniature car in the virtual space, the lanedetectorM module takes input from environment simulator module CamGen (which produces virtual image data similar to a camera input during on-road conditions). Using data from CamGen and controlling commands input by the user in virtual space or using the test scenario model, the driverM module determines the current vehicle position. driverM moduel also calculate the demand velocity (V_d), desired steering wheel angle (θ_d) to be applied using inputs from lanedetectorM and driving instructions.

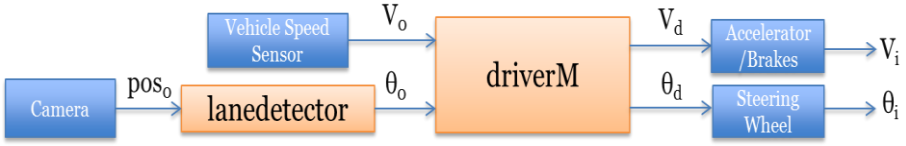

Fig 3: Miniature vehicle in running condition (open loop).

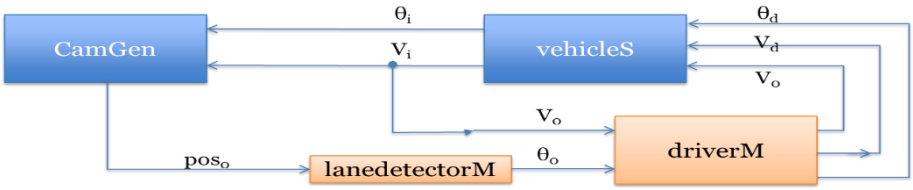

Fig 4: Vehicle in virtual simulation mode (closed loop).

The output of driver module is used to control the vehicle movement in case of on-track mode, or in virtual simulation is feed back to the vehicleS module to calculate the new vehicle position using linear bicycle model. The new position from vehicleS module is then used by CamGen to generate new image data and irus to re-calculate the obstacles distance to be used by lanedetectorM and driverM modules. Fig 3 and Fig 4 represents working mode in on-track and virtual simulation mode.

4.1. Injecting fault into the system

Now consider a simple scenario, we wish to simulate how the vehicle would act in case of faulty speed sensor (sensor output is zero). In the real vehicle/miniature vehicle on track, even though the speed sensor has failed at t=t0, we can reasonably assume that vehicle would continue in motion with initial velocity, v0 and process the observed camera images to navigate the lane according to lanedetectorM input. Although due to failure in vehicle speed sensor the vehicle speed would be assumed by driverM (system) as zero and thus demand maximum speed resulting in full throttle leading to vehicle accelerating and continuing operation in full speed mode.

But if we simulate the same condition in virtual space, the fault condition of zero vehicle speed would be interpreted in above described manner (like in real case) by the vehicleS model to simulate a condition with full acceleration demand, but the wrong signal (zero vehicle speed) will make incorrect new vehicle speed and thus also the distance traveled from the point of fault injection leading to faulty position interpretation by CamGen and thus the vehicle speed and trajectory in simulated case will not reflect the actual behavior and thus unreliable to make analysis.

Using simplified 1D model, the current velocity and distance can be calculated using Newton's law of motion,

$$v = v_0 + at$$

$$S = S_0 + v_0 t + \frac{1}{2} a t^2$$

In case of actual vehicle on track, due to faulty vehicle velocity input (v = 0 m/s) the driverM module will demand maximum acceleration (assumed here $a_{max} = 5\ m/s^2$), but the initial velocity irrespective of the state (working or faulty) will be v_0 (assumed below to be 40m/s) will follow the laws of motion. Also the observations from camera unit will be normal and thus the vehicle would be able to navigate the obstacles and follow the lane.

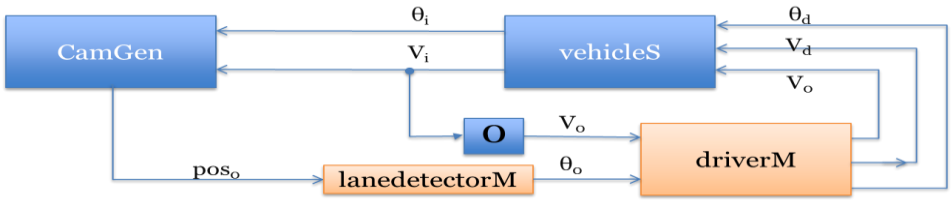

Fig 5: Vehicle in virtual simulation mode with fault injected.

While in case of simulated environment the initial velocity will be wrongly taken to be zero and although the driverM module will demand similar condition of maximum acceleration, in this case the simulated velocity and distance traveled would be wrongly calculated. And since in simulated case the module CamGen is used instead of real camera, the generated image based on wrong position data from vehicle module will result in faulty image generation thus vehicle would not navigate or follow lane correctly. Fig 6 shows the difference between velocity and distance in actual and simulated case.

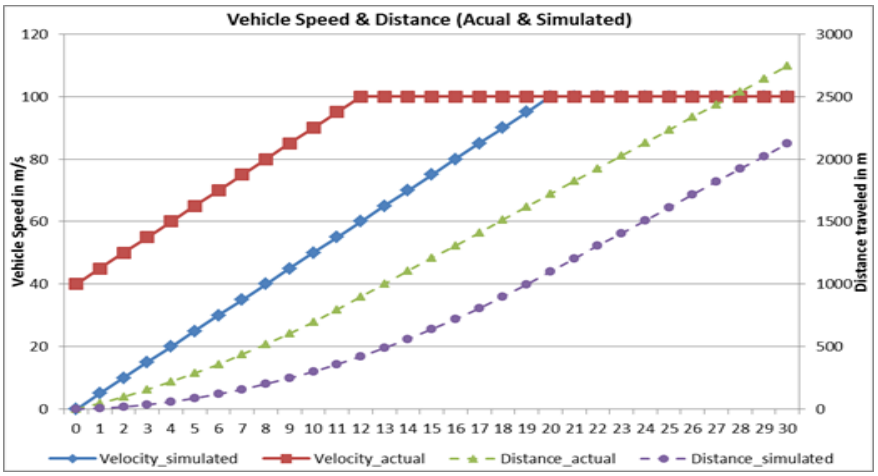

Fig 6: Velocity and distance traveled (actual and simulated).

Such inconsistencies occurs *due to dependencies and superficial feedback loops between the system and its environment where a system state/signal is used to calculate/control a natural parameter which in normal circumstances would not depend on that signal/ state of the system* [20]. In the given case study, the problem occurs due to virtual vehicle dynamics simulator (vehicleS) will take wrong input of current velocity (as zero) in fault scenario which is used to calculate the new velocity and new vehicle position, which is further utilized to generate the virtual image data by CamGen and thus producing incorrect simulated outcome.

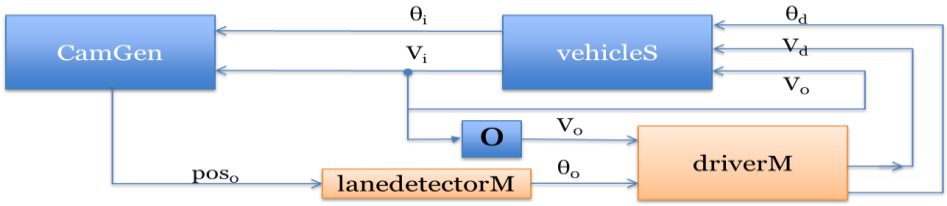

Fig 7: Vehicle simulation closed loop testing using FBM.

The solution for such problems is easily achieved by using principle of fault bypass modeling where the part of signal or its derivative, which is used to calculate/control the environment parameter (in this case correct initial velocity) is made fault free to break the unrealistic feedback loop. Thus in above case applying FBM principle, the initial velocity of moving vehicle is a parameter independent of injected fault, thus while to simulate the given fault scenario, fault (v=0) needs to be inputted to the driver module, but the fault free current value of initial velocity should be bypassed to the simulated environment (vehicleS module) so that the new velocity and position data is correctly generated and thus the output of CamGen (generated virtual image data). The implementation of FBM in given case is represented in Fig 7.

This is a simple example but for many embedded systems that require closed loop testing – transient properties are important or even critical. Consider testing for if the vehicle stops safely under scenario of failed brakes or how the pacemaker or some implantable device would react to an intermittent discharge from the battery. Using fault injection methodology to test for these fault scenarios under closed loop strictly depend

on ensuring that the system-environment simulated output is reliable and reflects the true behavior of system under test. Thus FBM principle outlined here can be useful for closed loop testing for dependability of non-deterministic systems and systems with high dependence on their environment.

5. CONCLUSION

We established that there is significant need for using closed loop testing of embedded software systems in many domains and applications. It is also discussed that fault injection can be used to enhance the effectiveness of closed loop testing by making it possible to do dependability evaluation of the system in early development stages. But injecting faults into closed loop configurations can generate outputs that are unreliable and may be unrealistic, to overcome this problem framework referred to as fault bypass modeling is demonstrated with a simple case study. Although the example discussed here is very simple, the use of closed loop testing is most often needed for testing of safety critical applications where dependability and reliability is of utmost importance thus FBM can prove to be a useful tool in ensuring dependability of embedded systems.

ACKNOWLEDGMENTS

The work presented here has been funded by Vinnova and Volvo Cars jointly under the FFI programme (VISEE, Project No: DIARIENR: 2011-04438).

References

[1] P. Liggesmeyer and M. Trapp, "Trends in embedded software engineering," *Software, IEEE*, vol. 26, pp. 19-25, 2009.
[2] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, pp. 42-52, 2009.
[3] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty, "Model-integrated development of embedded software," *Proceedings of the IEEE*, vol. 91, pp. 145-164, 2003.
[4] B. Graaf, M. Lormans, and H. Toetenel, "Embedded software engineering: the state of the practice," *Software, IEEE*, vol. 20, pp. 61-69, 2003.
[5] G. Buttazzo, "Research trends in real-time computing for embedded systems," *ACM SIGBED Review*, vol. 3, pp. 1-10, 2006.
[6] R. Van Der Straeten, T. Mens, and S. Van Baelen, "Challenges in model-driven software engineering," in *Models in Software Engineering*, ed: Springer, 2009, pp. 35-47.
[7] S. R. Dalal, A. Jain, N. Karunanithi, J. Leaton, C. M. Lott, G. C. Patton, *et al.*, "Model-based testing in practice," in *Proceedings of the 21st international conference on Software engineering*, 1999, pp. 285-294.
[8] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Improving Fault Injection in Automotive Model Based Development using Fault Bypass Modeling," in *GI-Jahrestagung*, 2013, pp. 2577-2591.
[9] R. Svenningsson, J. Vinter, H. Eriksson, and M. Törngren, *MODIFI: a MODel-implemented fault injection tool*: Springer, 2010.
[10] L. Kuzniarz and M. Staron, "On Practical Usage of Stereotypes in UML-Based Software Development," in *Forum on Design and Specification Languages*, Marseille, 2002, pp. 262-270.
[11] M. Staron and W. Meding, "Using Models to Develop Measurement Systems: A Method and Its Industrial Use," presented at the Software Process and Product Measurement, Amsterdam, NL, 2009.

[12] D. Trawczynski, J. Zalewski, and J. Sosnowski, "Design of Reactive Security Mechanisms in Time-Triggered Embedded Systems," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 7, pp. 527-535, 2014.

[13] Z. Jiang, M. Pajic, and R. Mangharam, "Model-based closed-loop testing of implantable pacemakers," in *Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems*, 2011, pp. 131-140.

[14] L. Stockmann, D. Holler, and D. Spenneberg, "Early simulation and testing of virtual ECUs for electric vehicles," in *International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium (EVS26)*, 2012.

[15] E. Bringmann and A. Kramer, "Model-based testing of automotive systems," in *Software Testing, Verification, and Validation, 2008 1st International Conference on*, 2008, pp. 485-493.

[16] C. Berger, M. Chaudron, R. Heldal, O. Landsiedel, and E. M. Schiller, "Model-based, composable simulation for the development of autonomous miniature vehicles," in *Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative M&S Symposium*, 2013, p. 17.

[17] W. F. Van Der Vegte and I. Horváth, "Achieving closed-loop control simulation of human-artefact interaction: a comparative review," *Modelling and Simulation in Engineering*, vol. 2011, p. 24, 2011.

[18] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J.-C. Fabre, J.-C. Laprie, *et al.*, "Fault injection for dependability validation: A methodology and some applications," *Software Engineering, IEEE Transactions on*, vol. 16, pp. 166-182, 1990.

[19] J. Vinter, L. Bromander, P. Raistrick, and H. Edler, "Fiscade-a fault injection tool for scade models," in *Automotive Electronics, 2007 3rd Institution of Engineering and Technology Conference on*, 2007, pp. 1-9.

[20] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Increasing efficiency of iso 26262 verification and validation by combining fault injection and mutation testing with model based development," in *8th International Joint Conference on Software Technologies-ICSOFT-EA, Reykjavík, Iceland, July 2013*, 2013, pp. 251-257.