Software-in-the-Loop Simulation for Early-Stage Testing of AUTOSAR Software Component

Sooyong Jeong, Yongsub Kwak, Woo Jin Lee
School of Computer Science and Engineering
Kyungpook National University
Daegu, South Korea
{kyo1363, iddqt, woojin}@knu.ac.kr

***Abstract*—In Embedded Software, the early-stage testing of source code is important since it may reduce the future development cost. However, at the automotive domain, the conventional software testing methods depend on actual system and hardware such as Hardware-in-the-Loop simulator and vehicle, then the early-stage testing method of automotive software is still in immature status. Therefore the testing of automotive software component is delayed until they are developed enough to run at the actual hardware. In this paper, we propose a method to do dynamic analysis and test the AUTOSAR software component by Software-in-the-Loop simulation without target system hardware. It provides rapid prototyping environment to validate the behavior of automotive software and helps to improve the quality of software component from the early error detection.**

***Keywords*— AUTOSAR, validation, Software-in-the-Loop simulation**

I. INTRODUCTION

In the automotive industry, more and more automotive electronic system and software are being used for the innovative functionality such as convenience and safety[1]. Cars in late 1970s used little software such as ignition timing control and simple trip computers. After that, the use of automotive software has been increased gradually by its demands. As a result, premium cars in 2009 include the ADAS, ESC, and so on, and it contains over 70 ECUs and close to 100 million lines of code[2][3]. Not only the premium cars but also economy cars contains a lot of code. Basically many functionalities depend on software such as steering, braking, engine control which directly related to safety of passengers and pedestrians. In addition, more and more up-to-date technology is also being adopted to economy cars. For example, 2016 Hyundai Avante, compact car, begins to contain the smart cruise control(SCC) and automatic emergency brake(AEB) system[20]. In short, the many cars are being run by the more software nowadays.

Consequently the needs to test the automotive software has increased, then the industrial and academic fields are researching the testing technology of automotive software such as AUTOSAR software component[4]. Until now, the many of the testing methods depend on hardware, but it is sometimes almost impractical and not suitable for early-stage testing. To test automotive software by conventional methods, the code should be compiled and built to executable for target hardware. It involves hardware configuration and porting to vehicle and ECU, so the testing processes should be done at the later development phase. And even though the software component is built to the executable for the vehicle and ready to be tested, the testing in practice is not so easy. For instance, in the case of the braking control software, it has branches about the road and traffic conditions, so it should be tested in various situations that makes branches of the system. However, reproducing the test scenario in real environment is difficult. For example, the shape and friction of roads are almost fixed environments and cannot be changed by testers' requirement easily. Similar troubles appear in the other parts in the vehicles such as powertrain and chassis.

In this paper, we propose a simulation on Software-in-the-Loop, to validate AUTOSAR[5] software component code at early-stage with less cost and risk. To test source code before the configuration of system and hardware, VFB layer of AUTOSAR is simulated in PC. The simulated layer is used to show the behavior and interaction of software components without hardware such as ECU. To validate the simulated software component with virtual prototype, the vehicle dynamics simulator is integrated to complete simulation and testing environment.

By the proposed simulation, AUTOSAR software component code is completely simulated and validated on PC realistically. In Section II we explain the development methodology of AUTOSAR and various automotive software testing methods in market, and show the needs of proposed method. In Section III the simulation processes to validate the software component are discussed. In Section IV a case study of validation with proposed simulation is shown, also it is presented that how the failure is caught and fixed. Finally this paper is concluded in Section V.

II. RESEARCH BACKGROUND

A. AUTOSAR Software Component Development Methodology

AUTOSAR methodology[6] defines the software and system development processes into the activities. They are classified to

This study was supported by the BK21 Plus project (SW Human Resource Development Program for Supporting Smart Life) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (21A20131600005), Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) [No. 10041145, Self-Organized Software platform(SoSp) for Welfare Devices], and the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the C-ITRC(Convergence Information Technology Research Center) (IITP-2015-H8601-15-1002) supervised by the IITP(Institute for Information & communications Technology Promotion)

5 subdirectories. Figure 1 shows brief AUTOSAR development methodology.

When a developer begins to make an automotive software using AUTOSAR, the first step is the system configuration. It includes modeling of software components, and network between components in Virtual Functional Bus(VFB), and so on. Then the models are represented to a document called System Description in ARXML format. Developers implement internal behavior of application software component from System Description, and make a source code of software component. Also System Description is used to the mapping to ECUs and subsystems in the target system, then it provide ECU Extract as a result. Basic Software(BSW) is a part which provide abstraction of ECUs and hardware. It contains operating system, communication, device drivers, and so on. Basic Software is generated by commercial generation tools. Finally, the products from respective processes are integrated to the executable software for specific ECU and deployed to vehicle and parts.

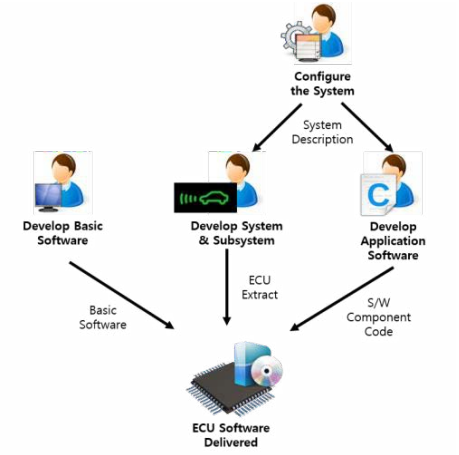

Fig. 1. Brief AUTOSAR Development Methodology

B. Automotive Software Testing Methods

As the importance of automotive software rises, a variety of testing methods for automotive software are developed and available at marketplace. In the practical software developments, methods using executable such as simulation, actual vehicle, and Hardware-in-the-Loop prevails in the automotive software testing[7]. Testing method based on actual vehicle and environment is essential to find the fault appears at software component in the reality. However, the testing is done usually at the later development process. To test software component at the actual vehicle, the software component under test should be executable in ECU. Also the reliability of the software component should be ensured before the testing, as the failure in the testing can makes serious accident. Because the driving is involved in the testing, the test scenario reproduction, and using the extreme test scenario is a hard job. Due to the limitation, Hardware-in-the-Loop is being used to test automotive software component instead of actual vehicle. Although it needs less complex test environment, the Hardware-in-the-Loop testing environment still requires the executable for ECU. To make the executable from source code, additional procedures like ECU configuration and basic software configuration are needed. Therefore the conventional methods are not appropriate to early-stage testing.

As an alternative, testing methods using virtualization and simulation are proposed. One of the methods is ARTEC-VFBS[9]. It generates virtual testing environment from AUTOSAR design documents. It supports simulation of some of electronic devices like clusters in Windows GUI and test the software component. Then it is available to early-stage testing, but it has critical shortcomings. Its simulation and testing coverage is limited to interior devices, and it uses Microsoft ActiveX, the deprecated technology. The other solution is ECU virtualization. dSpace VEOS[10][11], and ETAS ISOLAR-EVE[12] are the popular testing tools based on ECU virtualization. They are providing precise testing technology, and no required additional hardware. But the developer and tester need additional knowledge of hardware such as virtualization of ECU, sometimes the new member should be hired and educated to use the testing tools.

From this survey, we got that the most of test methods for automotive software are done in the later development process. However, the software faults may appear at every development phase, and a fault may be propagate as the development progresses[8]. That's why the method to detect and remove faults from automotive software in early-stage is mandatory. So we made a research to fulfill the needs that shows lightweight early-stage testing method for automotive software component. It is expected to contribute to reduce extra development cost from fault propagation, and help to improve potential software quality and reliability.

III. Simulation-Based Testing Method of AUTOSAR Software Component

To test AUTOSAR automotive software component before the hardware configuration, we propose the method that ports the automotive software component code to a Software-in-the-Loop simulator running in PC. Figure 2 shows the proposed method in rough. AUTOSAR software component under test is run on VFB simulated by API of PC. Simulated VFB acts as an interface between AUTOSAR software component codes with PC. Code under test combined with simulated VFB is compiled into a PC application called Code Simulator. Code Simulator expresses the communication and behavior of simulated automotive software components in PC. The Code Simulator is integrated with the vehicle simulator to mimic the physical effects from software components' behavior. As a result, the virtual prototype of vehicle to test the software components is established at the PC. Then software developer and tester can test the software component by the simulation.

A. *Code Instumentation by Simulating VFB*

AUTOSAR software components are designed over Virtual Functional Bus(VFB), then it is mapped into Runtime Environment(RTE) and Basic Software(BSW) which are dependent on hardware[13]. This layered architecture makes the AUTOSAR software component can be developed in parallel with hardware and system. Although RTE and VFB are fundamentally different concepts, they are designed to share common AUTOSAR Interface[13][14]. So Application software components can interact with other software component or hardware via AUTOSAR Interface in any development phase.

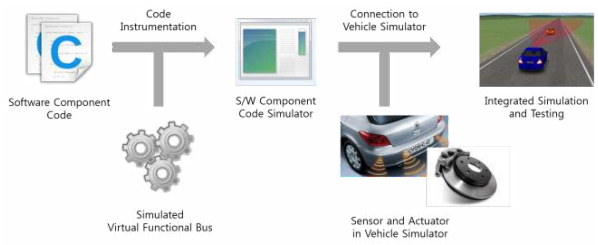

Fig. 2. Procedures of the Proposed Simulation-Based AUTOSAR Software Component Testing

AUTOSAR system and its software components usually consist of multiple ECUs and distributed environment. It involves multiple RTEs and BSWs, and that has a difficulty to implement the Software-in-the-Loop simulation because of hardware abstraction and operating system of respective ECUs. To minimize the overhead of simulation, we proposed the simulation technique based on simulating VFB. As VFB concept is centralized, less complex, and hardware independent, simulating VFB is a simpler approach. Because VFB use the common AUTOSAR interface used by software component, the unchanged software component code can be tested over the simulated VFB also.

For hardware-software co-simulation, there are various simulation techniques[15]. In this paper, our VFB simulation partially adopted the idea of Honda[16] with the arrange. The functional models for software components are provided that connects application tasks and host computer by management functionalities. After that, software components and simulated VFB are compiled by host computer's compiler together and executed by host computer. The simulated VFB is constructed as Figure 3. The simulated VFB comes from Software Component Description documents and is written in C code. To represent inter/intra ECU communication, the port connections and interface for software components are implemented. Shared memory is used to data transmission and communication. As most of operations are event-driven, event handler is implemented to simulated VFB. It handles timing events, and the events by other operation, etc. Scheduler manages the timing of system and invokes components' timing event that execute runnable. Finally, the entry point and control loop of simulation are located.

The simulated VFB with software components code are compiled and built to executable of host computer. This executable is called Code Simulator which simulates software component code and the closed loop without sensor and actuator.

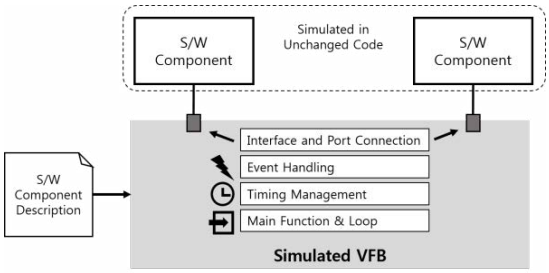

Fig. 3. The Proposed VFB Simulation

B. *Virtual Environment Integration to Code Simulator*

In general cases, automotive software interacts sensors and actuators of vehicle, then it makes the actions. That's why actual vehicle and Hardware-in-the-Loop simulator are used to test the automotive software in many cases. Especially if the software component directly connected to driving, validating on the actual vehicle is inevitable. However, testing faulty and immature software component under development at actual vehicle can be dangerous.

We propose the way that connect the Code Simulator to existing vehicle simulator to build an environment to validate software component, instead of actual vehicles. Figure 4 shows the connection. To link the vehicle simulator with Code Simulator, simulated VFB is extended for the vehicle simulator. The header for vehicle simulator API is included at the simulated VFB. The sensor and actuator of the vehicle simulator are initialized to establish the connections with the Code Simulator. The connections are used in the control functions of the simulated software component. By the connection, when the software component calls the control function via AUTOSAR Interface, the sensor/actuator in the vehicle simulator works instead of the actual vehicle. The main function of Code Simulator is synchronized to the timer of vehicle simulator and mimics the elapse of time.

As a result, the extended Code Simulator works as a Software-in-the-Loop simulator which simulate software components' effect on the vehicle in trustworthy way without an actual vehicle.

C. *Testing Simulated Software Component*

Using the proposed Software-in-the-Loop simulator, software components are running in the virtual prototype of vehicle by the simulation, and the behavior of software components is reflected to the action of the virtual prototype. Then the simulator can be used to validate the behavior of software component from the action of virtual vehicle.

The input for the vehicle simulator consists of 2 elements. The first one is procedure which represents roads, traffics, driving, and so on. The other one is vehicle configuration that includes performance and parts. Vehicle configuration is supposed to done before the creation of test scenario. The tester create a test scenario for the software component from the code, specification document, requirement, and so on[17][18]. It is represented to the procedure of vehicle simulator and its expected result.

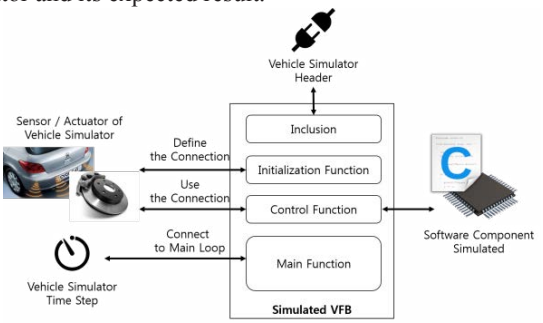

Fig. 4. Extension of the Proposed VFB Simulation Technique

After the creation of test scenario, the proposed Software-in-the-Loop simulator run the scenario with vehicle configuration. After the simulation terminated, a log file is generated by the vehicle simulator. The tester examine whether the log file satisfies the expected result in the test scenario. If the log was satisfied with expected value, the software component is regarded that passed the test. Otherwise, it means that the software component failed the test, and it implies that there is a fault.

With the proposed method, the AUTOSAR software component can be validated about its requirement in early-stage by Software-in-the-Loop simulation without the target system and system configuration of software.

IV. Case Study on the Proposed Method

To show the usefulness of proposed simulation-based validation method, we applied it to a practical development processes to validate the requirement and improve its functionality. An automatic emergency braking system is simulated by our method in the practice, and the simulation contributed to improve the system's quality.

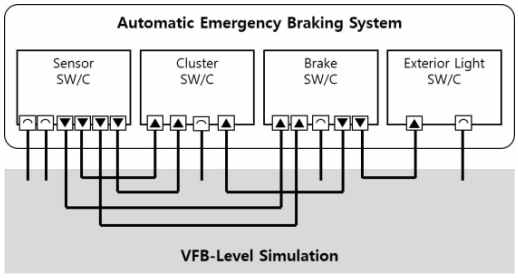

Fig. 5. Software Component Diagram of Automatic Emergency Braking System

Figure 5 shows the component diagram of automatic emergency braking system that is applied to our simulation method. The software system consist of 4 software components and communication with sender-receiver interface. The software components developed by Vector DaVinci Developer and customization. Sensor software component collects data from front object sensor and speed sensor measuring the distance and relative velocity to the nearest object throughout the driving. The collected sensor data are sent to cluster software component and brake software component. Brake software component read the sensor data and control the brake master cylinder automatically to avoid the collision to an object. If brake software component activate the braking automatically, it notifies cluster and exterior light that the braking is activated.

The automatic emergency braking system is required to guarantee of collision avoidance at fast velocity and every road condition. However, the validation of software component in conventional method is very hard. First of all, loading software component to the vehicle for testing should be done at late development step because vehicle and its ECUs can run executable format only. In addition to this, getting the required road conditions at every testing in reality is almost impossible. In contrast to this, the proposed method enables to simulate the software component without the hardware and its configuration, and control test variable such as road condition easily. So it supports early-stage validation of automotive application software component.

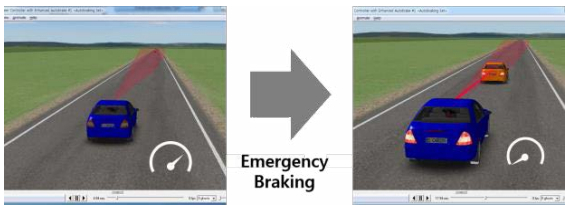

Fig. 6. The Simulation of Automatic Emergency Braking System

We built a Code Simulator by simulating VFB into C code. To simulate VFB, System event, alarm, and task are simulated by Win32 Thread, and Event. The shared memory in PC mimicked port to port communication. CarSim by Mechanical Simulation[19] is used as virtual prototype and environment and linked to Code Simulator. Test scenarios are written in CarSim procedure which expects the software component in development prevents the collision at certain velocity and road condition. By the test using the scenario, we had found the fault from the software component and tried to resolve them.

Figure 6 shows a validation example of automatic emergency braking system. A CarSim procedure is made to represent a test scenario consists of a sedan driving in highway situation. The car discovers an obstacle such as stopped car while the driving at 80km/h, and the brake is activated automatically by the sensing values and avoid a collision. Varying the road friction according to the condition, we expect the system avoids a collision at the extreme condition and

detect faults. If faults were detected by the simulation, the revision and testing were repeated.

Figure 7 illustrates the velocity graphs to validate the automatic emergency braking. The expectation was (i) the velocity of vehicle should be zero before sensing distance become zero, (ii) the braking should be successful in the extreme condition such as snow. At the early version of automatic emergency braking, the system didn't avoid the collision at the snowy and icy road. Also, the braking curve was steep and it means the car pitched harshly. From the result, we revised the code so that recognize the object in longer distance and brake earlier. As a result, the system was improved to cover the snowy road and less pitch at ordinary conditions.

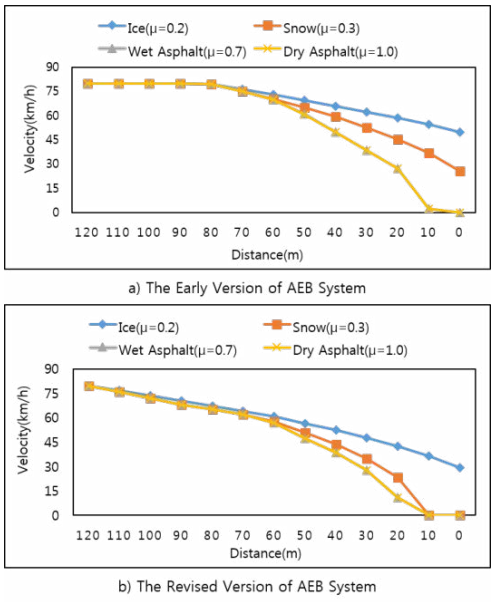

Fig. 7. The Braking Curve by the Automatic Emergency Braking System under Test

From this experiment, it is shown that the proposed approach is usable to validate a functionality of software component practically in the early stage. It has been efficient to validate the behavior of application code solely, and the testing at extreme situations, that is hard to be tested in reality.

V. Conclusion

In this paper we discussed about strength and weakness of existing automotive software testing methods and claimed the needs of early-stage testing method for automotive software.

Being inspired from AUTOSAR methodology and its layered software architecture, we proposed a validation method based on Software-in-the-loop simulation. The proposed method uses an idea that a software component source code to be ported to a PC executable which simulates its behavior by simulating of the Virtual Functional Bus and AUTOSAR Interface. Then integrating the simulated executable to commercial vehicle simulator, we got a realistic simulation. It supports running and testing automotive software application code without target system, so it supports early-stage validation unlike conventional testing methods. As a case study, we showed that the proposed method helped to the automatic emergency system software be validated and reduced its fault.

We expect the proposed method contribute to the various area of automotive software development and testing, and automating the validation process.

References

[1] K.Grimm, "Software Technology in an Automotive Company - Major Challenges," in Proceedings of the 25th International conference on Software Engineering, pp.498-503, 2003.

[2] M. Broy, "Challenges in Automotive Software Engineering," in Proceedings of the 28th international conference on Software engineering, 2006, pp.33-42

[3] R. N. Charette, "This Car Runs on Code," in IEEE Spectrum, vol. 46, no.3, 2009

[4] K. Sung, T. Han, "Development Process for AUTOSAR-based Embedded System," International Journal of Control and Automation, Vol. 6, No. 4, 2013.

[5] AUTOSAR. [Online] Available :http://www.autosar.org

[6] AUTOSAR Methodology. [Online]. Available:http://www.autosar.org

[7] H. Altinger, F. Wotowa, M. Schrius, "Testing Methods Used in the Automotive Industry : Results from a Survey," in Proceedings of the 2014 Workshop on Joining AcadeMia and Industry Contributions to Test Automation and Model-Based Testing, pp. 1-6, 2014.

[8] L. Baresi, M. Pezze, "An Introduction to Software Testing," Electronic Notes in Theoretical Computer Science, vol.148, no.1, pp. 89-111, 2006.

[9] J. A. Rowson, "Hardware/Software Co-Simulation," in the Proceedings on 32st Conference on Design Automation, pp. 439-440, 1994.

[10] ARTEC-VFBS. [Online]. Available: http://www.esm.co.jp/service/autosar/

[11] dSpace VEOS. [Online]. Available:https://www.dspace.com/en/inc/home/products/sw/simulation_software/offline_simulator.cfm

[12] Muli, M. and Cassar, J., "Virtual Validation - A New Paradigm in Controls Engineering," SAE Technical Paper, No.2013-01-2404, 2013.

[13] ETAS ISOLAR-EVE. [Online]. Available:http://www.etas.com/en/products/isolar_eve.php

[14] N. Naumann, "Autosar runtime environment and virtual function bus," Hasso-Plattner-Institut, Tech. Rep, 2009.

[15] W. Dafang, L. Shiqiang, H. Bo, Z. Jiuyang, "Communication Mechanisms on the Virtual Functional Bus of AUTOSAR," in the Proceedings of International Conference on Intelligent Computation Technology and Automation, Vol.1, pp.982-985, 2010.

[16] S. Honda, T. Wakabayashi, H. Takada, "RTOS-Centric Hardware/Software Cosimulator for Embedded System Design," in Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp. 158-163, 2004.

[17] C. Nebut, F. Fleurey, Y. Le Traon, J. M. Jezequel, "Automatic Test Generation : A Use Case Driven Approach," IEEE Transactions on Software Engineering, vol.32, no.3, pp. 140-155, 2006.

[18] W.T. Tsai, L. Yu, X. X. Liu, A. Saimi, Y. Xiao, "Scenario-Based Test Case Generation for State-Based Embedded Systems," in the Proceedings of the 2003 IEEE International Conference in Performance, Computing, and Communications, pp.335-342, 2003.

[19] Mechanical Simulation CarSim. [Online]. Available:https://www.carsim.com

[20] Hyundai Avante Official Site. [Online] http://www.hyundai.com/kr/showroom.do?carCd1=RD026#showroom-cont-1295