New Testing Process for Software Development in Hybrid Electric Vehicles

Mihai-Ovidiu Nicolaica
PhD Student, Faculty of Electronics, Telecommunications & Information Technology,
"Gheorghe Asachi" Technical University, Iaşi, Romania

ABSTRACT

The present paper introduces a new testing process for embedded software development activities in the field of hybrid electric vehicles. The proposed process addresses some deficiencies of the existing testing models in order to improve the performance of HEV embedded software testing activities. The value added is given by the clear split of the test tasks and the level of independence between the test vehicles involved in the test process. The motivation of the research is given by the fact that the automotive sector is heading towards electric vehicles and the level of software complexity in these vehicles is continuously increasing. The advantage of this method is that will allow a smoother offload of testing tasks between locations, depending on project needs. The test model contributes to the practical need of developing software for hybrid and electric vehicles at higher quality standards, faster and with a better cost.

Keywords

HEV; testing process; software

1. INTRODUCTION

1.1 Software in HEV

When comparing to the conventional vehicles, there are more electrical components used in electric, hybrid, and fuel cell vehicles, such as electric machines, power electronics, and embedded powertrain controllers. Advanced energy storage devices and energy converters, such as Li-ion batteries, ultra capacitors, and fuel cells, are introduced in the next generation powertrains. In addition to these electrification components or subsystems, conventional Internal Combustion engines (ICE) and mechanical and hydraulic systems may still be present in the vehicles [1]. Most innovations within the automotive domain are driven by embedded systems and software solutions. The costs for embedded solutions in vehicles are growing rapidly. The costs for embedded solutions in vehicles grew from 1 percent in 1980 over 7 percent in 1990 to 22 percent in 2007. It is estimated that the importance of embedded systems and software in electric vehicles will grow much further. One major innovation within the automotive domain is the introduction of electric vehicles. Therefore, the embedded systems and software challenges in the hybrid electric vehicles become more difficult. Current electronic architectures consist of up to 100 electronic control units for hybrid electric vehicles. A good testing process is needed to handle this complexity. Today, embedded systems and software in HEV are essential for the competitiveness of the automotive industry. Their most notable effects are to improve driving performance, comfort, and to enhance both passive and active safety. Modern vehicles comprise dozens of spatially distributed embedded systems. These ECUs are often developed by different suppliers with most diverse requirements on safety, reliability, costs, and computational power. The integration of distributed components into a modern vehicle is a challenging task [2]. The software development becomes one of the most challenging activities in the field of hybrid electric vehicles being responsible of full integration and safety.

1.2 Software testing in HEV

In a few words, software testing represents an engineering activity performed in order to assess the software quality. It should be demonstrated that the software behaves as expected. The expected behavior is defined in the design documents and in the system and software requirements. A good testing process has the goal to detect errors with a minimum effort and in a minimum amount of time. Due to the increased complexity of the system and software in hybrid electric vehicles, testing gains more and more importance in the development process.

By testing, the engineering team can validate the product before release to market and in this way the quality and the confidence is increased. Currently the test activity starts after the code development, and requires as much time as the actual build of the code. Sometimes the time needed for complete testing the code exceeds the time the code was written. Also the costs of testing can exceed the development cost. This happens because, currently, there is a linear approach in the product lifecycle. There is a dependency between design, development and testing. More than this there is a dependency between the test levels involved and this affects the performance in terms of time and costs of the final product. The system and software design team, software development team and system and software test team should have the same starting point and work in parallel trough the development lifecycle. Of course, there will be still a task dependency at certain gates in the product development, but this should be kept to a minimum in order to reduce the overall development time.

The test teams should be involved, through a representative, from the design phase, in the requirement definition and customer interaction in order to assess the testability of the design intent to avoid scrap and rework. In this paper it is underlined the fact that in the field of hybrid and electric vehicles the way of improving the efficiency of the testing activities is to involve the test teams as early as possible. In the same time all test teams should fulfill their specific task in parallel. This approach improves the testing time, the accuracy of the results (because one test team does not relay anymore on the results of another team), and in this way increases the quality of the final product at a lower cost.

1.3 Importance of testing
The importance of testing has increased tremendously in HEV since the system and software have become more complex [3]. Efficient testing has a direct impact on the software quality and mainly on safety. The cost is also an aspect that needs to be considered. The cost of a failure increases dramatically if it is found in production or in service. A recall of a vehicle fleet can have a decisive impact on the business. This is even more problematic in case of hybrid electric vehicles, where the market is still fragile. Testing performed according to an efficient testing process can help avoiding underestimated project costs or quality escapes. Using the proposed testing process, it can be proven that the HEV software meets the ISO26262 quality standards and the corresponding quality levels. Hybrid electric vehicles have more and more components with ASIL D safety level and the manufacturer is responsible of the damages that happen to the product user and other road participants.

1.4 Testing levels
In general, in literature, the testing is structured in a set of four levels: component test, integration test, system test and acceptance test [4]. There can be defined also some intermediate levels, but this depends on the developed product type. For HEV, these levels are valid and the proposed test process is developed around them.

1.5 Testing types
There are different testing types that can be applied on different test levels. The testing activities can be divided in black box testing or white box testing, static testing or dynamic testing, manual testing or automated testing. The present paper underlines the importance of automating the test steps at every level. Test automation has the advantage of one hundred percent reproducibility and once created, the test scripts can be executed again on each software drop. This approach speeds up the testing activities and increases the reliability of the results. The proposed testing process underlines the importance of the test automation and encourages the usage of this approach.

2. DEVELOPMENT V CYCLE
2.1 Process Overview
The V cycle development model is currently the model used while developing software in HEV. It has certain advantages and disadvantages. A main advantage is the fact that the development and testing activities are considered with the same importance. The actual model has one branch dedicated to the software development and one dedicated to the testing tasks. A representation of the V cycle is visible in Fig. 1.

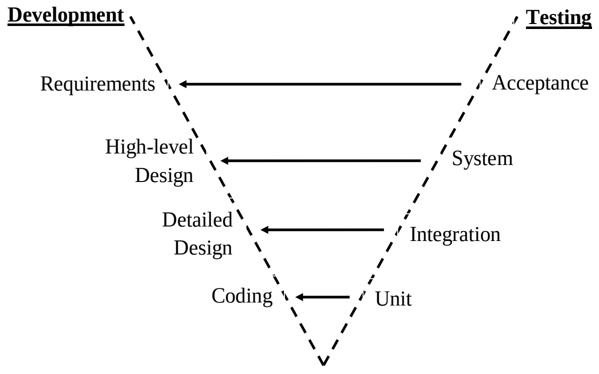

Fig 1: Development V Cycle

The V cycle is used in different versions and the names of the levels vary from one version to another. The main idea that needs to be understood here is the fact that the development and testing need to have granted the same importance. This is underlined also by the new proposed test process. Only in this way complex systems like hybrid electric vehicles can have the expected level of quality.

2.2 Cross industry assessment
Currently, the complexity of HEV software is similar to the software available in the aviation industry or biomedical equipment. From certification perspective are clearly different but the level of complexity and quality constraints are raising in the automotive market. The levels of safety and accuracy are more and more demanding. The proposed test model is intended for automotive industry, but due to its portability and adaptability can be easily translated to other industries.

3. Testing HEV software
3.1 Actual testing models
There are different testing models or processes in the current automotive industries that help developing high quality software. Due to the evolution of software quantity and functionality the existing processes need to be also updated. The current direction in the automotive industry, especially HEV, is to develop new technologies faster, to a higher quality and with a better cost. Cost cutting and task offload represent another aspect that needs to be considered in the actual economic environment. In order to keep up with the market tendency, the test processes need to be aligned, and this is what the current paper tries to support.

Actual HEV products are developed based on the test model visible in the V cycle or using waterfall model, spiral model or scrum. All of them have a linear approach where each test vehicle follows a previous one. If an issue is found at the end of the test process there is a need of a complete reiteration of the entire process. This means time and resources blocked to a certain degree. The proposed testing process tries to eliminate the linearity and dependency between test vehicles. If a reiteration is needed than this can be done only by the team that is required, freeing up resources that have nothing to do with the corresponding finding.

3.2 Deficiencies and improvement proposals in actual test processes
Lacks in the actual testing process are: test vehicle dependencies, linearity of tasks, and full reiteration of the testing process in case of failure.

Possible improvements could be: parallel testing approach, independence of testing, easier offload of test tasks, cost, and time to market.

4. PROPOSAL FOR NEW TESTING PROCESS
The current paper proposes a new testing model that tries to address some shortages in the existing testing processes. The proposed testing model is focused on improving the testing accuracy and independence.

4.1 Overview (Rhombus Testing Process)
In order to get valuable test results there is a need of a certain degree of independence between testing vehicles. The proposed approach focuses on requirements based testing in order to avoid testing based on specifications. Testing based on requirements assures different points of view for certain

functionalities. The requirements should be unambiguous, atomic and clear. The tester has a certain degree of freedom in defining the test steps based on his understanding of the functional behavior.

The intention of the proposed model is to have different specific requirements for each testing level (low level testing, hardware in the loop testing, system testing and acceptance testing). The requirements are flown down from top to bottom, starting from the customer requirements, as specified in the development V cycle. There is a strong emphasis on the traceability that is considered very important in this case, together with the test coverage on all levels. The new proposed testing model is named rhombus or diamond model. The main idea is to assure independence between the testing vehicles through the life cycle of the software drop. A representation of the proposed testing process is presented in Fig. 2.

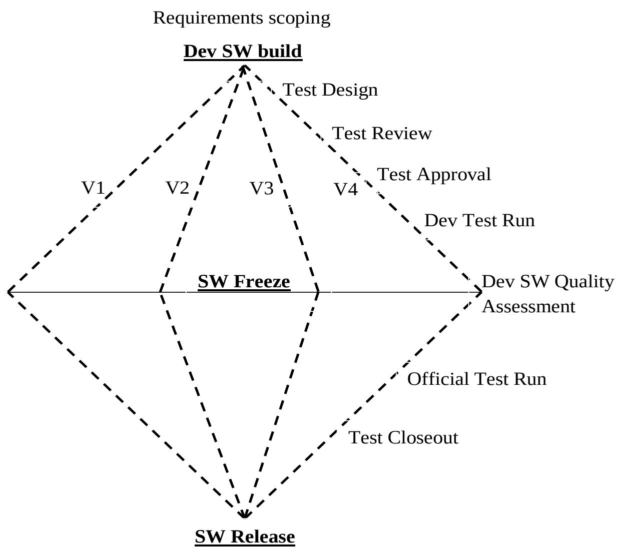

Fig 2: Rhombus testing model

There is a common starting point where the work split is agreed together with the requirements trading. There might be the case that some high level requirements cannot be tested on a high level and in this case low level testing needs to support. Also, sometimes, a system level effect needs to be assessed for some low level requirements. The new testing model proposes only three common points for the involved test vehicles. A common point is when the testing activities start. This is intended to concentrate on planning activities. A second common point is defined after the last development build. This is intended to assess the software quality in the middle of the development cycle and decide if the software can be frozen. For the software freeze decision, all involved parties need to agree and all test results need to be analyzed. If the outcome of the analysis shows that the software implementation is at the expected quality standard the implementation can go formal. The third common point is at the end of the testing activities, performed with frozen formal software.

4.2 Process levels

The proposed process involves different layers of testing with similar testing gates. The testing steps are valid for each test vehicle and are executed in parallel by each of them through the lifecycle of the software drop.

The defined levels of testing or test vehicles are:

- Low Level Test – V1
- Hardware in the Loop Test or Integration Test – V2
- System Test – V3
- Acceptance Test – V4

The defined test gates for each test vehicle are:

- Requirements scoping
- Test Design
- Test Review
- Test Approval
- Development Test Run
- Development SW Quality Assessment
- Official Test Run
- Test Closeout

The proposed levels of testing and the corresponding test gates can be customized based on the project needs and defined at the beginning of the project in the master verification plan. It might be that another test vehicle or another test gate needs to be added. Before the software development starts all test vehicles need to be involved in requirements review. The review must focus at this moment in time on the testability assessment. Every requirement must be verifiable at each level. After all requirements pass the testability assessment than software can start implementing the functions in the code. The work split between the test teams must be defined before the initial development software build is available. The software team can work independently on building the code based on the reviewed requirements and when ready the compiled version can be uploaded in a configuration management tool. From this moment in time there is a clear split of responsibilities and task independence. Each test team representing the test vehicles V1 to V4 can take the initial code and initiate the testing activities. The starting point is the test design activity. The test engineer will define a test strategy composed from a set of test steps derived from the system or software requirements. Depending on the verification level the test steps can turn into manual or automated tests. Different tools can be used, and these are specific to each team. If the test are automated than a set of tests scripts are developed. It is recommended to have a high degree of test automation. The level of automated test scripts should be at least 85% and should be monitored accordingly. Before execution, the test steps and test scripts need to be reviewed by another engineer. This exercise validates the test strategy and increases the quality and confidence in the test process. After the review process is passed and the testing approach is approved the testing can be initiated.

Each level can provide test results on the initial development build. Before the test results are communicated to the software team they need to be again reviewed by another engineer. It can be the same engineer that reviewed the test strategy and scripts or another one. Each test level can raise issues found while testing the initial development build. In the same time a software update can be triggered by findings from any level. If for example low level testing reports one or more issues and the software is decided to be updated, than it will be configured in the change management tool and will be available also for hardware in the loop test or system test. Each test team needs to continue the work with the last software version available in the database. This approach forces an immediate reaction at every code change from the test teams. Based on the test findings the embedded software can be updated several times. There is not defined a specific number of development builds. The decision when the last build is done comes from the project plan. When development testing is stopped all test vehicles need to discuss results. Open findings are analyzed at each level and a common status needs to be provided. Based on the result it is decided if the software freeze can be done or not. If an important software issue is still visible at this level, and if other test vehicles and software team agree, there can be another development build. The decision of freezing the software brings all test teams together for the first time since the start of the test process on the development builds.

Once decided that the software can be frozen, independent testing can again continue until the decision of releasing the software will be made. After the software is frozen, it needs to be configured again in the configuration management tool. In principle, this version should be the final one and no change is intended anymore. Having this final version, all test teams need to reconfirm the test scenarios and results on the frozen software code. This activity can be defined as official test run. At this moment the test cases are stable and the results should confirm the conclusion drawn during development coding and testing. Any issue spotted at this moment in time would reiterate the development process and the code should be unfrozen. Here the advantage of the proposed testing model would be that only the level that reported the issue could re-execute the development testing. After the official test run it is done each test level needs to provide a test report or a test closeout. All reports are then discussed in a quorum where all verification levels meet for the third time in the testing process. Based on the final results a decision is made if the software can be released. The proposed testing process is designed for baseline testing or for testing independent software drops with only certain fixes that address correction of existing functionalities. Based on the level of the changes the number of the involved testing teams can be adjusted.

4.3 Advantages and improvements

The main advantage of the proposed testing process is that it assures a level of independency between the test vehicles. Each team (low level testing, hardware in the loop testing, system testing, acceptance testing) follows the same process steps but without having a task dependency between them. The only dependency to progress the testing tasks is on software drops and software quality. This avoids unplanned delays in the testing activities, caused by other testing team’s inability to accomplish deadlines. Another aspect is that the test tasks can be done in parallel by separate teams distributed in different locations, all over the world. Adopting the proposed testing process will allow a smoother offload of testing tasks between locations, depending on project needs.

5. CONCLUSION

The present paper proposed a new testing process that is intended to be applied during software development for hybrid and electric vehicles and is based on a newly defined Rhombus testing process. A short analysis of the actual state of the automotive industry with regards to testing in HEV was presented. Some improvement opportunities were identified and some solutions were presented trough the proposed test process. Due to its portability the process can be easily implemented also into other industries like aviation or medicine. The proposed model tries to improve the software development approach in order to increase the software quality, development time, resource allocation and planning. Using this model the control over the project lifecycle will be increased and the task offload capacity will be improved.

The testing steps are described on a high level. The difference between software testing for HEV and other safety critical embedded systems are not significant. The paper is written in line with HEV for validation purpose in order to demonstrate the value of the proposal. The link towards HEV is intended to demonstrate how it would be applied in practice. A concrete set of testing results are not available since the proposed testing model represent a conceptual approach. It is intended to be a process definition for testing embedded systems. One of the main points is to have different specific requirements for each testing level (low-level testing, hardware in the loop testing, system testing). The new process relies on performing different levels of testing in parallel, contrary to current practice (V model) which suggests running test levels one after the other. The intention is to make testing more efficient. Applying the prosed test process has more value added in a context where the code works already reasonably well. This process can be applied with better results for software updates rather than brand new software development.

6. REFERENCES

[1] D. Wenzhong Gao, C. Mi, A. Emadi, "Modeling and Simulation of Electric and Hybrid Vehicles", Proceedings of the IEEE, Vol. 95, No. 4, April 2007

[2] S. Chakraborty, M. Lukasiewycz, C. Buckl,S.b Fahmy, N.k Chang, S. Park, Y. Kim, P.k Leteinturier,H. Adlkofer, " Embedded Systems and Software Challenges in Electric Vehicles", EDAA, 2012

[3] M. Raetzmann, C. De Young, "Software Testing and Internationalization", Lemonine International, Inc. Salt Lake City, 2003

[4] A. Spillner, T. Linz, and H. Schaefer, “Software Testing Foundations” Rocky Nook Inc. Santa Barbara,4th Edition, 2014.