Vehicle Level Continuous Integration in the Automotive Industry

Sebastian Vöst
ISTE, University of Stuttgart
Universitätsstrasse 38, Stuttgart, Germany
sebastian.voest@informatik.uni-stuttgart.de

ABSTRACT

Embedded systems are omnipresent in the modern world. This naturally includes the automobile industry, where electronic functions are becoming prevalent. In the automotive domain, embedded systems today are highly distributed systems and manufactured in great numbers and variance. To ensure correct functionality, systematic integration and testing on the system level is key.

In software engineering, continuous integration has been used with great success. In the automotive industry though, system tests are still performed in a big-bang integration style, which makes tracing and fixing errors very expensive and time-consuming. Thus, I want to investigate whether and how continuous integration can be applied to the automotive industry on the system level.

Doing so, I present an adapted process of Continuous Integration including methods for test case specification and selection. I will apply this process as a pilot project in a production environment at BMW and evaluate the effectiveness by gathering both qualitative and quantitative data. From the gained experience, I will derive possible improvements to the process for future implementations and requirements on test hardware used for Continuous Integration.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification—*Validation*; D.2.5 [**Software Engineering**]: Testing and Debugging; K.6.3 [**Management of Computing and Information Systems**]: Software Management—*Software Maintenance, Software Process*

General Terms

Management, Verification

Keywords

Automotive, embedded, testing, continuous integration

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08...$15.00
http://dx.doi.org/10.1145/2786805.2803193

1. INTRODUCTION

Within the last decade, embedded software systems have been on the rise more than ever. Nowadays, microprocessors can be found everywhere in our daily life. The automobile industry is no exception to this.

Quite the contrary, it is a domain which has seen excessive growth when it comes to functionality based on software. In a modern car, software is present in a unprecedented diversity with a range from engine control systems to navigation and entertainment systems. These systems are spread over up to 100 individual control units with very different specifications, many of which are involved in a singular function simultaneously. Summed up, the software integrated in a vehicle is a conglomerate of highly distributed embedded systems with very distinct requirements.

The same can be said about the avionics industry, but unlike airplanes, cars are a mass product and are often produced and sold in numbers of hundreds of thousands. To make things worse, today's customers expect high customizability in their vehicles, so that control units and software components alike exist in different variants, each offering different functionality. Many of these can be combined, so in the end, several hundred different configurations are sold in a single product line. This implies a different approach to the development, but even more so to the testing process compared to other domains.

1.1 Problem Statement

In software engineering, continuous integration has been adopted by many software projects and has made regular system testing a common sight. This technique originates from Extreme Programming and applying it essentially means that the software is fully integrated and tested after every change. This reduces risks in software projects, most notable in this case, the late discovery of defects [1].

Yet, system integration in automotive embedded software is performed in a rather late stage today. Integrations of several software components that run on the same control unit might be performed earlier, but with regards to the high distribution and variance, system integrations become more and more important.

This is not without reason, though. As of now, frequent integrations are hindered by several problems.

Due to the proximity to hardware, the test cases need to be run on the actual control units. Thus, test cases take a very long time to execute in comparison to pure software tests and require significant effort to be run automatically. In addition, we are facing organizational problems, because

the development of software for a vehicle can be considered an extremely large project with hundreds of developers distributed over dozens of teams in different companies.

Even more problematic is the design of test suites. Running tests on the hardware makes them expensive, so we have to select the tests that are actually relevant. Effectively, we have to design the test suite dynamically based on the tested change. The same applies to the selection of variant combinations, on which tests are executed.

To guide the selection, we have to define goals that we try to reach in every integration we perform so that we can select the test cases and variants most suitable. As of now, there is no clarity about which goals are at all practical and which of these provide the best results. We will also have to apply these goals to the specification of test cases to make sure the necessary test cases are available.

The tests would finally be executed on special hardware, so-called test benches that represent a certain part of a vehicle with respect to electronic components. We have to take these test benches into account and figure out what requirements they have to fulfill with regard to flexibility and automation.

1.2 Research Objective

The overall objective of my thesis is to investigate how the application of continuous integration on the vehicle level can be made possible in the automotive domain. This includes the adaptations of the process itself, as well as test case design, selection, variance management and hardware requirements.

1.3 Context

I have been working on my thesis internally at BMW in cooperation with the University of Stuttgart. Thus, I am involved in internal projects that aim to improve the integration process, will have access to exemplary productive data and I am able to perform empirical studies investigating the practical application of the presented concepts.

2. RELATED WORK

Papers on the application of continuous integration in the automotive industry in particular are scarce. However, the broader field of agile methods and embedded systems in general has been investigated quite thoroughly.

In their paper, Shen et al. [5] have investigated about 50 studies regarding the adoption of agile development techniques to embedded software development. They found that the majority of methods applied originated in XP, which is the origin of continuous integration. They did not explicitly differentiate between the specific XP techniques though. Overall, they concluded that agile development is not at a very mature stage in the embedded domain, but that the majority of developers noticed a positive impact. None of the papers they investigated had an explicit focus on the automotive domain, however.

One of the major problems in the automotive industry is the organizational size and the distribution of developer teams. With regard to continuous integration, this problem can be found similarly in very large software projects. Indeed, Roberts [4] has proposed a modified approach called Enterprise continuous integration (ECI) that can be applied to the automotive domain. ECI is composed of multiple stages, in which integration results from small teams are later integrated with binary dependencies developed by other teams. Since system integration in the automotive industry is usually done with pre-compiled software components, and developer teams or suppliers often perform their own integration anyway, a similar process seems suitable.

Some research in the automotive industry has been performed by Richenhagen [3] on continuous integration of powertrain systems. In his dissertation, he proposed a framework to continuously integrate embedded software in the automotive domain. However, Richenhagen took an approach from the developers' point of view, concentrating on low-level integration and did not take a partial or full system level into consideration. Besides, he focused on powertrain components exclusively. Some of his ideas could be transferred to a more holistic approach, however.

We can find lots of interesting approaches to the design of test suites in the automotive industry in a paper by Lachmann and Schaefer [2]. They cover the entire testing process, from test case specification to test selection and tackle many concrete problems specific to the automotive industry. For example, they describe an interesting solution to streamline test cases written in natural language, which is very common in the domain. Other methods to provide a structured test suite include highly automatic test selection based on machine learning or redundancy removal based on comparable test cases. Lachmann and Schaefer address black-box testing as done in system testing, however they do not take a continuous testing approach into account, but their research is rather based on the prevalent strategy, in which tests are executed only after certain development steps have been reached. Also, they do not provide any detailed algorithms that can be reused, but many of their ideas provide a good starting point for my own research.

3. APPROACH

Follwing up, I will present my approach to achieve my general research objective and derive more specific research questions from this approach, since a major part of my work is the empirical evaluation of the developed concepts.

3.1 Terminology

First of all, let us clarify the used nomenclature. In the introduction, I used the term *Integration*. In the context of this work, an integration includes every step performed automatically after a software change has been made. This may include static code analysis, compilation of binaries and flashable packages, flash tests and the execution of test cases for verification. An integration in this context does not include manual tests or crash tests. I will use the terms *Test Platform* and *Integration Platform* as synonyms. Both refer to a certain type of test bench that is able to run the software under test and execute certain test cases on it. The test bench is a well-defined part of a full system (a vehicle) and physics and hardware are either fully (Software-in-the-Loop) or partially simulated (Hardware-in-the-Loop). A *test suite* in this context is part of an integration. It contains a set of test cases and a number of hardware variants on which these have to be performed. A test suite always refers to a certain test platform.

3.2 Automotive Continuous Integration

If we want to focus testing on continuous system testing in a domain, where no holistic concept of continuous in-

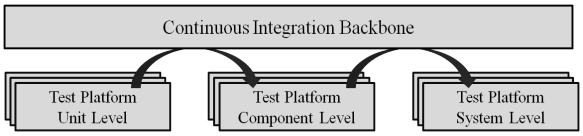

Figure 1: Staged Continuous Integration

tegration exists, we first have to create a clear context in which the different testing levels can be investigated separately from each other.

Thus, we split the involved parties in a *Continuous Integration Backbone* and several different levels, as shown in Fig. 1 with three exemplary levels.

There can be several test platforms linked to the backbone on each level, so long as they provide a given interface which specifies input and output artifacts for each stage.

In the vehicle stage, the input artefacts would be precompiled software containers specific to a certain ECU. These can be delivered by an earlier stage which produces these containers as output. Since we assume the system level to be the last stage, our output artefacts do not contain any software but merely the reports and evaluations of executed test cases.

The backbone triggers action on a certain level when a more recent version of any input artefact is delivered either externally or as result of a previous level. The backbone will generate one or more test suites for the affected levels and distribute the tasks over the associated platforms.

To prevent a permanent blockade of the integration server by broken commits, we let go of the traditional notion of *breaking the build* as it is. Whenever a change causes an error on some stage, the build will not simply fail and cause all following changes to inevitably fail as well until the error is fixed. Instead, the change will be rejected on all levels and the responsible team will be notified. Following builds will be performed based on a revision prior to all possibly erroneous changes.This means in fact, that a software version can always only be finally accepted into the repository of a stage when it has successfully passed through all stages.

I want to investigate whether or not this process is appropriate, so the first research question is:

- RQ1: Is a staged and modular CI process effective and applicable in the automotive industry?

3.3 Test Suite Design

This process would rely heavily on individual integration platforms that automatically generate their test suites based upon a change. Prerequisite for this is a large set of test cases that are specified to achieve the best results in an integration. However, the nature of these results, the *integration goals* need to be defined. These may vary for each stage, but in this case we want to focus on the system stage exemplarily.

Coverage criteria like code coverage are often used as metrics for the quality of a testing environment. On a system level, we do not have access to the source code and assume that this aspect has been checked adequately in previous stages. However, we can define similar criteria.

On the system level, the artefacts that we want to check are the user-visible functions. These functions are described by a number of requirements. Only if every requirement of a user-visible function is covered by test cases, we can decide whether or not the function is working correctly after an integration was performed. Thus, the first suitable criterion seems to be *requirements coverage*.

Variance coverage could serve as a second criterion. The problem of a high number of variations in ECU's and therefore their software was introduced in section 1. A focus of the system level is to detect faults in the communication between two components in a distributed function, so we need to ensure that the function was tested in all relevant hardware variants.

Finally, the general goal should be to maximize *error detection*. This criterion rather aims at a continuous improvement rather than a first-time-right approach. Error detection can be maximized e.g. by analysis of errors that were found in other test instances (like manual testing) or even in serial operation. Regression tests can be derived from such errors and adapted to components that are prone to similar errors.

- RQ2: Are Requirements Coverage, Variance Coverage and Error Detection suitable criteria to drive test case specification and test suite design?

The effort for a full integration will likely increase to a level where it is hard to impossible to maintain within a continuous integration process and with a realistic budget. In any case, a high degree of automation will be necessary, but this might not suffice.

- RQ3: How can appropriate coverage of these criteria be achieved in a continuous integration process? Is this possible at all within a realistic budget?

3.4 Integration Platforms

Finally, we want to investigate if there are existing test platforms that can support this process. At BMW, there are several candidate test benches that could be used as test platforms for different stages. These include a highly automated Hardware-in-the-Loop concept that allows the electronic components of a whole vehicle to be run in a simulated environment. Thus, virtual driving is possible in the laboratory and test cases can be run fully automated.

- RQ4: Can Continuous Integration be implemented on existing platforms in the automotive industry and what are possible improvements for future platforms?

3.5 Research Plan

I plan to finish my dissertation by December, 2017. Within the first year, I will be performing most of the planning work. I am designing the continuous integration and test suite design processes and will gather possible partners to implement them in a pilot project. The second important field I have been and will be working on in the first year is to enable a test platform to support a continuous integration process. For this, I am designing a framework for automatic control of a test platform which will be implemented and operative by early 2016.

In my second year, I will spend most of the time rolling out the designed processes. The goal is to set up a project in which a couple of software development teams as well as integration platforms on different stages take part. During this time, I will collect first data about the technical and organizational difficulties one encounters when implementing such

a process at an original equipment manufacturer (OEM). Also, I will be able to collect quantitative and qualitative data about the efficiency and acceptance of the automated testing framework.

In my final year, I will mostly do evaluation of the implemented processes. I will collect data on test runs, achieved test coverage and code quality where possible to answer my research questions and write up the results.

4. EXPECTED CONTRIBUTIONS

The general contribution of my work is an adapted continuous integration process for embedded systems that can be practically applied by OEMs in the automobile industry.

In order to evaluate its effectiveness, I will design two frameworks implementing it. One general integration backbone, which handles changes, distributes them to appropriate test platforms, interprets test results.

The second one is a test platform manager. The framework's purpose is to automatically generate the test suite for its designated test platform based on a given changeset and subsequently execute them on the associated hardware. The test suite generation includes test case selecetion aiming for maximum coverage of the criteria referenced in RQ2 und RQ3 within a limited timeframe.

These frameworks are necessary to conduct the studies that will allow me to estimate whether or not the proposed process and the driving criteria are valid, they can be covered appropriately and to learn about imposed requirements on the executing hardware.

5. PLAN FOR EVALUATION

As mentioned before, I am working on my dissertation as a part of BMW's own research projects, so I will have many opportunities to gain empirical results.

I will support the design and establishment of a company-wide continuous integration process and evaluate it in the form of a case study. For this, I will select several appropriate integration platforms and software development teams. In a qualitative analysis using interviews with participating developers and integrators, I want to provide insight on the acceptance of the process and find strengths and weaknesses for further improvement.

To estimate whether or not the selected criteria are valid, I intend to measure the error escape rate and compare it with traditional means of guided testing. In addition to this, I will review the achieved test results with experts in order to uncover possible gaps.

I will try to judge whether or not the process is effective by a quantitative analysis that takes into account the average time needed for integrations and the overall satisfaction of the coverage criteria. If the criteria are not met sufficiently, I will try to extrapolate the remaining effort to answer the second part of RQ3.

If possible, I will perform a qualitative and quantitative analysis on software quality. Possible metrics include the comparison of the number of detected errors over the different development phases with continuous and traditional integration.

I will perform a similar case study with a focus on the adaptation at a certain stage. This part will have a more technical nature and I will try to figure out how well the requirements of the continuous integration process can be applied by the integrating business units. This study particularly aims to identify the requirements on the hardware in use.

6. RESULTS SO FAR

Based upon several previous works including Roberts' [4], I proposed the process of automotive continuous integration [6], which allows a modular implementation of the process on independent stages.

Quantitative or qualitative results cannot be presented yet. However, the general response from developer teams at BMW is positive so far and some teams are very eager to participate in a pilot project as proposed.

7. CONCLUSION

System level integration in the automotive industry is a challenging task and imposes many problems that are not very well researched yet, at least not in an automotive or embedded context. The lack of papers on the application of continuous integration strategies in the automotive industry clearly indicates that this field of research needs to be investigated more thoroughly.

With this work, I want to provide a first approach to a holistic continuous integration for automobile embedded software. This approach will be developed and established at BMW such that I will be able to provide empirical results first-hand from a realistic and operative environment.

At the end, I aim to provide to the community a better insight on how a modern and concrete technique like continuous integration can be applied to the complex field of embedded automotive systems as a holistic integration approach.

8. REFERENCES

[1] P. M. Duvall, S. Matyas, and A. Glover. *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[2] R. Lachmann and I. Schaefer. Towards efficient and effective testing in automotive software development. In *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern*, pages 2181–2192, 2014.

[3] J. M. Richenhagen. *Entwicklung von Steuerungs-Software für den automobilen Antriebsstrang mit agilen Methoden*. dissertation, Rheinisch-Westfälischen Technischen Hochschule Aachen, 2014.

[4] M. Roberts. Enterprise continuous integration using binary dependencies. In *5th International Conference, Proceedings, Extreme Programming and Agile Processes in Software Engineering, XP 2004*, pages 194–201, 2004.

[5] M. Shen, W. Yang, G. Rong, and D. Shao. Applying agile methods to embedded software development: A systematic review. In *SEES 2012, Zurich, Switzerland. Proceedings*, pages 30–36, 2012.

[6] S. Voest. *Use of concepts from continuous integration in the automotive industry in the context of e/e-development*. Master's thesis, Technische Universität München, 2014.