Embedded Automotive Software Development in Electronic Modules for Commercial Vehicles

Caio Luiz de Carvalho Ferreira
Ivan Arantes Levenhagen
Lucas Gomes de Almeida
Volkswagen Truck and Bus

ABSTRACT

There is a lot of uncertainties about how an automotive company can start developing their own software. What technologies should they use, what tools and resources? Is it possible to use agile methodologies in this type of development?

We will not deep dive in standards and complex metrics, since there are a lot of documentation that deals with this but give an overview of difficulties and paths to be followed.

This paper describes the embedded software development, tailored for commercial vehicles (trucks and buses) from its foundation and feature definition to its final release to production line, and tries to help other teams to take the best decision when discussing about bringing this responsibility to their companies.

INTRODUCTION

For many years, VWTB (Volkswagen Truck and Bus) bought electronic modules from their suppliers with the software already developed and embedded in the supplier factory. The process was simple. Along with the hardware specification, the VWTB engineering team sent the software specification, describing its functionalities and behaviors. The final product was the electronic module and its software sent as a "black box". Our teams knew how it should work, but not how it was written.

This kind of strategy has some drawbacks in the new technologic and agile scenery that we are living. Its common that when a software modification is requested to support a new legislation or a new feature, the costs and the timing provided by the supplier exceeds the client budget and may compromise the project deadline.

Based on those arguments we decide to change our culture and strategy bringing software developers to make this development in-house and, with that, reduce our costs, time and incorporating this development know-how inside the company.

METHODOLOGY

There are some factors that makes this development more complex.

One of them is the safety, that must be widely discussed before this implementation. When developing software for vehicles, we must have in mind that we are also handling human lives. There are international standards [1] that help us with safety[2] and security[3] metrics for better hardware development, but we will not cover that in this article.

Other factor it's the local legislation where vehicles with this software will run. If the software will be used only in one country or will be exported and be subject to a new set of rules completely different.

This makes that the process of developing an automotive software be strictly documented and audited in all its phases, to ensure quality, safety and compliance with legislation. In the next steps, we present the standards used in the development lifecycle.

IATF 16949 - In VWTB we are audited in accordance with IATF 16949, which is a quality certification for the automotive sector and, within this certification, software development requirements are foreseen.

A-SPICE – to ensure quality and satisfy all documentation requirements to this kind of project, we adopt as reference the Automotive SPICE, a standard of best practices and development maturity assessment, with a validation and verification model based on V-Model.

The V-Model consists in two stages, development and testing. For each development phase there is a testing phased

1 ISO 26262.

2 Safety: measures for protecting the environment from the automotive.

3 Security: measures for protecting the automobile from the external environment and intrusions.

related. After all coding, the software is tested in a sequential order foreseen at the beginning of the project.

A-SPICE covers all phases of a software development, since the project managing and requirements gathering to the final release, through coding and testing of the entire project. It has a compendium of base practices and outcomes for each development phase, and it is a complete guide through all this journey.

SCRUM - Although embedded software still follows a traditional waterfall development model, due to all its requirements and documentation phases, we wanted to bring an agile approach to our projects.

"Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value. In a nutshell, Scrum requires a Scrum Master to foster an environment where:

1. A Product Owner orders the work for a complex problem into a Product Backlog.
2. The Scrum Team turns a selection of the work into an Increment of value during a Sprint.
3. The Scrum Team and its stakeholders inspect the results and adjust for the next Sprint.
4. Repeat. " [1]

DOCUMENTATION – Since the start of the project, we had to write all the software specification, the features and requirements that our trucks and buses would have to attend. We create a requirement sheet model, and every functionality was written using the same model. This structure makes easy the understanding of everyone involved in the project, and it's the base to writing the software test use cases.

FAILURE AND RISK ANALYSIS – It's very important that every functionality has an analysis of possible risks and failure that the software is subject. For this purpose, it was used FMEA[1].

"FMEA is a step-by-step approach for identifying all possible failures in a design, a manufacturing or assembly process, or a product or service." [2]

NETWORK ARCHITECTURE – Electronic modules are not alone in a vehicle architecture. There are a lot of other modules that has to communicate between themselves. The most common protocol used in automotive industry is the CAN protocol. A network architecture must be created to provide this communication, and we decided to use a hybrid architecture mixing the "Bus and Star topologies" to create an optimized network. This hybrid concept brings us some advantages:

1. Medium cost due to the complexity and the necessity to have one brain ECU.
2. Wiring harness optimized due to mix of Bus topology and star topology.
3. High scalable, possible to add new ECUs due to Bus topology.
4. Multiple CAN Bus allow the system to be more robust.
5. High flexibility, since it is best of each architecture.

DEVELOPMENT STRATEGY – To simplify the software structure, we decided to develop one unique software for all vehicle models, and configure different behaviors through different parameters for each one. This kind of development makes a cleaner and generic codes, that can be reused in different applications. It's a balance between complexity and maintainability.

Aiming future developments and possible code maintenance, a modular programming strategy was thought to facilitate this process. Each developer in the team is responsible for coding a set of functionalities for the electronic module, defined in the beginning of the project.

There is a person called *integrator*, that receives the unitary codes for each developer and integrate all functionalities in an entire and unique software project.

The developer is responsible for the initial testing and simulation of their own code, ensuring that it has no compiling problems in the software, such as infinity loops, stack overflows and undesired interruptions.

SECURE DEVELOPMENT – A set of best practices that ensure the software quality from its concept to the final release. These are some of the best practices for secure development:

1. Think security from the beginning;
2. Create secure software development policy.
3. Design software with best practices to meet security requirements.
4. Protect code integrity.
5. Review and test code early and often.
6. Be prepared to mitigate vulnerabilities quickly.
7. Configure secure defaults settings.
8. Use checklists.
9. Remain agile and proactive.

PARAMETRIZATION – At the development phase, there is a support team that configures and parametrizes different vehicles applications that the test team will run. For the final release phase, all vehicles models and configurations are provided by an automated system that runs in the production line.

1 FMEA: Failure mode and effect analysis.

TEST STRATEGY – According the development best practices, it's strictly not recommended that the developer makes the functional tests for their own code. Based on that, there is an entire test team, separated from developer team, responsible by writing the test cases for each unitary functionality and test it in bench and in the vehicles.

DEVOPS – Its widely used in software development for I.T. applications, but its concept can be applied to any development project. DevOps can coexist with Agile software development; IT service management frameworks, like ITIL; project management directives, such as Lean and Six Sigma; and other strategies.

A DevOps culture means developers get closer to the user by gaining a better understanding of user requirements and needs. The word DevOps is a combination of the terms *development* and *operations*, representing an approach to the tasks performed by the company's development team and other operational teams.

The key premise behind DevOps is collaboration. During our documentation process we invite experts in other areas involved in the truck development to help us writing better specifications and aggregate knowledge to the team.

DevOps it's not a technology, but a general concept that apply a set of methodologies common in the software development area. These include the following:

Continuous Integration and Continuous Delivery (CI/CD) - Are practices that speed up the software releasing process by shortening feedback loops and automating repetitive tasks;

Software healthy monitoring – Real-time monitoring of software releases, incident management and collaboration platforms;

Cloud computing – microservices and containers / virtual machines implemented concurrently with DevOps methodologies;

TOOLS AND RESOURCES

After defining the standards to be followed and the methodology to be used, it's time to define the tools that will support all the development.

PROGRAMMING LANGUAGE – the programming language chosen to our development was Matlab/Simulink. It is a visual language, in blocks and easy to understand. Software programming using visual elements such as blocks or mathematical models is known as MBD (Model-Based Design).

The Matlab/Simulink platform has a lot of toolbox that aggregates value to the software. It can analyze the software for potential security breaches, infinity loops and integration missing links. It has the possibility to automate the software compilation and simulation in a cloud system and use the CI/CD functionalities provided by Azure DevOps.

VERSION CONTROL – also known as source control, is the practice of tracking and managing changes to software code. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

The system used to version control was GIT, a powerful open source platform used widely in the programming world.

INTEGRATED PLATFORM - To support this development, with all its particularities, we needed a platform capable of serving as a point of convergence for all our requirements. We needed a place to store all the documentation for all phases of A-SPICE, with an easy-to-understand structured view and a repository for all the code to be developed.

In partnership with our Information Technology sector, we acquired Azure DevOps, a Microsoft platform with high customization power, where it is possible to store all our documentation and control the versioning of all our code, in addition to allowing the creation of plans of tests, test automation and full reports on the health of our software.

THE ENTIRE PROCESS OVERVIEW

This section provides a summary of the entire process using the methodologies and tools presented in this paper.

First of all its created a scope off all functionalities and legislation that the new vehicles have to attend, following the A-SPICE flow of development. After that, the engineering team defines the hardware and its specifications to support the new features. It is informed to the possible suppliers the safety level (ASIL) the module has to attend, according ISO 26262 and the programming language that the module has to be compatible.

Defined the hardware, the engineering team writes the list of functionalities to be implemented in the software. We call those "functions". Each function has to be analyzed according FMEA and its requirements must be written by a team of engineers of the areas involved. Each function has to be presented to a forum with all the areas for acceptance and approval. That the time to make adjustments in the functions and change some requirements.

It is started then the V-Model flow using the Scrum approach. The project owner defines the complexity of the

1 FMEA: Failure mode and effect analysis.

functions, the priorities and the delivery time for each one. The functions are distributed to the developers in a called "sprint planning" and its defined the deadline for its deliveries. Every day the developers makes the "daily meeting" to report the progress of activities. All software modifications and new implementations are send to the version control system and are integrated in the final software by the *software integrator*.

At the end of each sprint, the team reunites to check if all activities were accomplished and plan the next sprint. Any problems encountered during the sprints are reported to the product owner, that tries to eliminated all possible barriers to the development process.

After the end of each development sprint, its released a new software version for the test team validate. The product owner reports to the test team what are the new features or bug fixes presented in this new release that needs validation. This new software is sent to the test area and configured by a support team, that realizes the correct parametrization for each vehicle model and family. The tests are realized in a bench and in the vehicle and if it has no known problem, the test team generates a report granting permission to follow the flow.

With the first test approved, the software is subjected to quality validation. For a few weeks, the vehicle will run in the streets and roadways to validate the full integration between all electronic modules inside the architecture. If the software passes this test, it is generated a report granting permission to be released to production line.

Any problems encountered are reported to the developers team, via Azure platform and are inserted in the next sprint planning for bug fixes If it's a concept problem, the process retrocedes to concept phase and are analyzed the impacts in the entire project.

All decisions, reports, specifications and any other outcome documentation are obligatory stored in the Azure platform in the documentation repository.

CONCLUSION

It is a very hard and challenging decision to develop your own software inside your company. That a lot of variables and questions that we do not have the proper answers at the beginning. But the advantages on developing your own software instead of buying it from other vendors outweighs the disadvantages of this process.

This brings cost and timing advantages. You can be more flexible with dates and deadlines and every scope changing can be quickly replied and absorbed in your chronogram. The application of new technologies and methodologies depends only on you and your team. In summary, is a great opportunity to evolve and create software development know-how within the company.

REFERENCES

[1] Scrum.org Press (2022, January 10). What is scrum? Scrum.org. https://www.scrum.org/resources/what-is-scrum

[2] ASQ Quality Press (2022, May 01). Failure Mode and Effects Analysis. ASQ. Https://asq.org/quality-resources/fmea