Otomotiv Sektöründe Genişletilmiş Çevik Yazılım Geliştirme Süreci Deneyimi

Fatih Öztürk¹, Koray Bilgin¹, Cem Sezer¹, Seda Güneş¹
¹ Red Pine Software R&D Centre,
İzmir, Turkey
{fatih.ozturk, koray.bilgin, cem.sezer, seda.gunes}@redpine.com.tr


Anahtar sözcükler: SAFe, ADAS, V-Model, Gömülü Yazılım, Yazılım Süreç İyileştirme, Çevik Yöntemler

Scaled Agile Software Development Process Experience in Automotive

Fatih Öztürk¹, Koray Bilgin¹, Cem Sezer¹, Seda Güneş¹
¹ Red Pine Software R&D Centre,
İzmir, Turkey
{fatih.ozturk, koray.bilgin, cem.sezer, seda.gunes}@redpine.com.tr

Abstract. The complexity of embedded software in automotive industry is rapidly increasing in parallel with the customer requirements in Advanced Driver Assistance Systems (ADAS). In order to fulfil the emerging requirements, Agile Software scaling frameworks have been adopted to automotive industry. This provided better frequency and predictability of releases in large projects. However, developing safety-critical systems with industry-scale agility is still a big challenge for distributed teams. In this study, we have identified the difficulties and key points that we experienced while applying V-Model for functional safety in a Scaled Agile driven global organization.

Keywords: SAFe, ADAS, V-model, Embedded Software, Software Process Improvements, Agile Methodologies
1. Introduction

In recent years, the use of various Agile methods in different organisations has been increased, while the use of Waterfall methodology is decreasing [17]. Previous studies have indicated the successful adoption of Agile methods in embedded software projects [18] together with others. The continuous development and test-driven mindset characteristics of embedded software development are compatible with the Agile principles.

Agile methodologies provide flexibility for the engineering teams especially in large-scale embedded projects. Scrum is the most preferred method among the various Agile methods such as XP (Extreme Programming), Kanban and Feature Driven Development (FDD). Scrum method does not describe reporting, source control or coding but it provides better predictability and risk management by using iterative and repetitive models.

Nowadays, Advanced Driver Assistance Systems (ADAS) are getting more critical as more applications are expected on premium vehicles including various forms of autonomous drive. Some safety functions are also becoming mandatory like Electronic Braking System (EBS), increasing the demand for all automotive range. Safety critical functionality is the major factor in complexity of the ADAS application development. In order to provide high-quality software components with the safety critical functionality, some specific procedures are required. In addition to the functional requirements safety requirements must also be fulfilled as described in the Functional Safety standard (ISO26262) [19].

The software in automotive industry is usually developed by multiple suppliers for single OEM (Original Equipment Manufacturer). V-Model is commonly adopted to automotive software development cycle. Suppliers are involved in the development process with a wide range of responsibilities such as application development, integration and testing. OEMs usually handle requirement specification, system design and final verification with acceptance testing.

The aim of this study is to give an insight and make a comparison for the use of Agile methodologies in the automotive software development process. In Section 2, some related work is available on applying Scaled Agile Framework (SAFe) practices for developing safety critical applications on AUTOSAR framework. In Section 3, background in automotive sector regarding Automotive SPICE, V-Model and ISO26262 are provided. In Section 4, our software development process experience with SAFe is presented. In Section 5 conclusion and future work are provided.

2. Related Work

Scaled Agile aspects and principles have been examined and presented [1] for some years now. Variety of Agile frameworks have been proposed [3] since the first models for scaling Agile in organizations was practised [2]. The common goal of these Agile frameworks is to increase predictability of the releases and the visibility across the whole organization. In order to achieve this goal in the entire enterprise, Agile mindset has been applied in different disciplines, such as Architecting, Design, Marketing, Portfolio Management or Program Management rather than only using it in software development team level [1].
Among Agile frameworks, SAFe has become the most popular and the most adopted one [4] for multi-site organizations who are scaling their Agile development processes. The growing trend for globally distributed companies to adopt Agile methods also emerged SAFe practices in global organizations [5].

In automotive industry, manufacturers and suppliers have been implementing complex applications for more than a decade, such as ADAS applications [7]. They produce safety critical software by following AUTOSAR specifications as an open standard for automotive electronics. The AUTOSAR standard provides modularity, scalability and reusability of functions. However, there is no formal verification of reliability and safety concerns in AUTOSAR. Car manufacturers address these concerns by building compliance to ISO26262 [19] in accordance to the Automotive SPICE model [22]. Plan-Driven Methods were used in early applications of the protocols in the automotive industry. Automotive manufacturers also adopted Agile frameworks to their development processes [20], although the conflicts between plan-driven processes and Agile approaches emerged in some early studies [11], [12], [13], [14].

Technological improvements rapidly transformed modern cars from mechanical entities to complex embedded systems. Industry players have continuously improved software engineering technologies to stay ahead of the game, as well as improving their underlying processes and practices. High-end cars currently have around 100 electronic control units (ECUs), running multi-milion lines of software spanning across safety-critical, driver assistance, comfort, and entertainment related applications [8]. Increasing complexity in software functionality emerged the need of distributed software and distributed development environments as the characteristics of automotive industry [9].

Agile practices and continuous delivery of software brought lots of advantages to automotive industry as well as in the embedded software world, although automakers lately started deploying Agile development. The biggest advantage is the efficiency in software development while maintaining the centralized decision-making necessary at the enterprise level. As the software complexity in the ECUs increase, there is need to address more sophisticated problems. SAFe provides the environment to mitigate these problems with a coordinated strategy for large-scale and complex projects with teams that number into the hundreds.

The key challenges to Agile adoption in the automotive industry involved transforming the organizational structure and culture, achieving a shorter release cycle without compromising quality [10]. Having this in mind, the paper presents a set of challenges relevant when Agile practices are scaled beyond a single team in organisations developing automotive software.

3. Background in Automotive Software

ADAS projects consist of complex software components and an underlying framework running on several ECUs. These components or framework can also be used in different car manufacturer projects. As depicted in Figure 1, ECU software system consists of more than one sub system aiming to serve different functionality such as lifecycle service and watchdog service. These services are considered as AUTOSAR compliant SWCs (Software Components) which are separated from the AUTOSAR basic
software stack [23] providing a layered software architecture via RTE (Run Time Environment) interface.

AUTOSAR stack brings us a layered architecture along with the requirements of the ADAS framework and having cross functional SWCs on different hosts with different ASIL levels defined in ISO26262 [21]. After collecting the requirements, system engineers have the responsibility to design the system so that hardware and software systems are compatible.

4 Scaled Agile Software Lifecycle Experience in Automotive

Heterogeneous and distributed software development on many automobile variants and configurations is the main characteristic of automotive industry. In the following three subsections we present some of the difficulties that we experience as a contributor company in this distributed environment. Subsection 1 and Subsection 2 give examples of improvements initiated by the sprint retrospective meetings which mitigated the negative effects on team's performance. Subsection 3 presents some actions taken as a result of the Problem-solving workshops in order to identify the ways to improve the team performance in the Program level of SAFe.
4.1. Agility in different levels of V-Model

There are several documents which are created and maintained during the development cycle according to the V-model process. These documents are linked to each other from any subsections to provide a chain from requirement specification to validation procedure. After capturing all the functional and non-functional requirements, SRD (Software Requirement Document) is created in Requirement Analysis Phase. These requirements are analysed further to design SWC functionality in unit level which is translated into SAD (Software Architecture Document) in System Design Phase. UDDs (Unit Design Document) are constructed at the same time as the developers write the code by following the SAD which is created in Subsystem Design Phase. Doxygen comments, function prototypes and global variable definitions and unit types are added in UDDs. Having all links from SAD to UDDs are available and aligned with each other, there are also UTS (Unit Test Specification) [15] documents consist of test plan and test procedure. Developers are responsible for adding unit test cases corresponding to the function which is tested.

Working with distributed teams in the traditional V-Model which is enhanced version of a waterfall model [16] would prevent improvements in development process. Making a change in the process would not be easy while working on the documents and implementing the related functionality. As a benefit of SAFe, our teams could interact with different levels and aspects of the system. For instance, it is discovered in a sprint retrospective meeting that a wrongly designed software component had been causing time loss during peer review process. A common unit would be better used and compiled for different hosts having different OS running on it, because that unit was not dependent on the target host. Our team took an action item and created a technical dept story to provide a solution. This story enabled the team to modify the architecture of the software component and avoided the unnecessary efforts spent for peer review process.

Table 1. Example review times after improvement

<table>
<thead>
<tr>
<th>Review Id</th>
<th>Component</th>
<th>Start Date</th>
<th>End Date</th>
<th>Work Hour</th>
<th>Review Type</th>
<th>Participants</th>
<th>Files</th>
<th>Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>3537</td>
<td>Persistency</td>
<td>13/06/18</td>
<td>24/10/18</td>
<td>120 hours</td>
<td>Formal Review</td>
<td>13</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>5106</td>
<td>Persistency</td>
<td>17/04/19</td>
<td>25/04/19</td>
<td>24 hours</td>
<td>Formal Review</td>
<td>12</td>
<td>9</td>
<td>21</td>
</tr>
</tbody>
</table>

As it is shown in Table 1, ideal review processes for the same component took less time with the similar set of participants, and this time includes both review and rework time. Although the complexity in the Review 5106 was higher than Review 3537, a considerable amount of time saved as a result of the improvement in the component.

4.2. Incompatibilities when reviewing code and document independently

The author is supposed to create a review task both for the document and the code after completing the code implementation. Ideally the document reviewer and the code reviewer should be the same person, but this is usually not the case in practice. Not having the same author in both the code review and the document review causes inconsistencies between the implementation and the documentation.
At earlier stages of the review process, code review and document review of the software units (UDD) was held independently. Once it has been addressed by our team in one of the sprint retrospective meetings, the team decided to combine the code reviews and the document reviews. This action accelerated the team velocity by saving a significant amount of time.

**Table 2.** Example review times for the same unit including code and UDD separately* and together*  

<table>
<thead>
<tr>
<th>Review Id</th>
<th>Component</th>
<th>Start Date</th>
<th>End Date</th>
<th>Work Hour</th>
<th>Review Type</th>
<th>Participants</th>
<th>Files</th>
<th>Defects</th>
</tr>
</thead>
<tbody>
<tr>
<td>4119*</td>
<td>Persistency</td>
<td>04/10/18</td>
<td>22/11/18</td>
<td>32 hours</td>
<td>Formal Review (Code)</td>
<td>15</td>
<td>8</td>
<td>23</td>
</tr>
<tr>
<td>4095*</td>
<td>Persistency</td>
<td>01/10/18</td>
<td>30/10/18</td>
<td>32 hours</td>
<td>Formal Review (UDD)</td>
<td>14</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>5110**</td>
<td>Persistency</td>
<td>17/04/19</td>
<td>03/05/19</td>
<td>28 hours</td>
<td>Formal Review (UDD + Code)</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 2 gives an example of the time spent for two different code reviews with ideal technical complexities. The total time spent for Review 4119 and Review 4095 is apparently higher than the time spent for Review 5110. The main reason behind the delay with the separate reviews is the number of defects had to be resolved back and forth in both reviews.

There is another type of incompatibility addressed by the team in sprint retrospective meetings. The root cause of this incompatibility was mainly the difference in coding rules between ASIL software components and QM software components. Since an ASIL D unit complies with the highest functional safety level functional safety awareness is required for a reviewer to complete review process for a higher degree ASIL SWC. If an engineer without enough experience on safety measurements is assigned to a review for a change in an ASIL D component, heavy delays occurs until the review is accepted and closed. As a result, another engineer with enough knowledge is mostly involved in the review.

4.3. Working with distributed teams

Working with distributed teams has always been a challenge in terms of management. The automotive industry is confronted with the management challenge of multi-tier outsourcing approaches. As a part of our business with other global organizations we also experience some impediments due to being geographically distributed. Our team identified the root causes of being a remote team in a Problem-solving workshop which is a part of Inspect and Adapt meetings. The team voted on the root causes to select the
most significant factor causing the problems in accordance with SAFe practices. In this subsection the selected root causes are described, and the selected action points are listed.

Time zone differences is one of the major reasons of the impediments. Arranging meetings can easily be a challenge due to the office hour differences. If you build up a scrum team meeting for a daily stand-up, you need to be careful to keep all team members comfortable enough to be available every day at the same time at the same place (a communication channel in this case).

Lack of face-to-face communication is another obstacle to building collaboration. The scrum team should be proactive, responsive and open to minimize isolation. Nowadays, conference and collaboration tools are quite enough to provide the environment for immediate contact between team members. However, scrum masters should spend extra effort to build such a team habit of quick calls.

Office culture can also vary in different countries and regions. Having different conventions can easily cause frustration between team members. Such as having more annual leaves, having different dress code, company benefits, flexibility in working hours and ease of access to some equipment are some example reasons for a possible frustration. As scrum encourages cross functionality, creating fair environments in different offices should be considered while forming distributed teams.

As face-to-face communication is more effective than text messages, building high collaboration between the distributed team members requires extra effort. In order to reduce the negative impact of not being co-located to other teams these actions are taken:

- Organizing workshops more frequently in different offices
- Scheduling onboarding trainings in headquarters
- Trying to form the teams co-located as much as possible
- Pairing senior engineers with new joiners
- Rotating team members between different teams

These actions improved the overall team performance especially in the onboarding process of the new engineers. Although the projects and the topics are highly variative in complexity, there is a considerable performance gain required for a new engineer to be operative on a domain. Initially the onboarding process used to take from 6 weeks to 3 months. After the improvements, the onboarding process rarely took more than 3 weeks.

5. Conclusion

In this study we presented our analysis on the use of agile and SAFe methodologies in the automotive software development. We identified some team level obstacles of being part of large-scale global organizations in automotive. Our experience with these obstacles is explained under several topics such as working with distributed teams, incompatibilities in reviews, agility in V-model. The study also explains how we coped with these problems by using V-model in combination with agile practices.

Having adopted SAFe practices in embedded automotive software development provided better agility and predictability, better organizational structures with huge teams, faster software release cycles without loss of quality. Such an environment allows engineers to have a better awareness of the complete picture in enterprise. Applying SAFe on highly complex software systems is sometimes formidable in
practice as we mentioned about the experience within third section. Although there are other agile frameworks which address this complexity, companies should adapt the agile framework in order to have best fit and value [3].

References


