Analyzing an Automotive Testing Process with Evidence-Based Software Engineering

Abhinaya Kasoju[a], Kai Petersen[*,b], Mika V. Mäntylä[c,d]

[a]Systemite AB, Gothenberg, Sweden
[b]School of Computing, Blekinge Institute of Technology, Box 520, SE-372 25, Sweden
[c]Department of Computer Science and Engineering, Aalto University, Finland
[d]Department of Computer Science, Lund University, Sweden

Abstract

Context: Evidence-based software engineering (EBSE) provides a process for solving practical problems based on a rigorous research approach. The primary focus so far was on mapping and aggregating evidence through systematic reviews.
Objectives: We extend existing work on evidence-based software engineering by using the EBSE process in an industrial case to help an organization to improve its automotive testing process. With this we contribute in (1) providing experiences on using evidence based processes to analyze a real world automotive test process; and (2) provide evidence of challenges and related solutions for automotive software testing processes.
Methods: In this study we perform an in-depth investigation of an automotive test process using an extended EBSE process including case study research (gain an understanding of practical questions to define a research scope), systematic literature review (identify solutions through systematic literature), and value stream mapping (map out an improved automotive test process based on the current situation and improvement suggestions identified). These are followed by reflections on the EBSE process used.
Results: In the first step of the EBSE process we identified 10 challenge areas with a total of 26 individual challenges. For 15 out of those 26 challenges our domain specific systematic literature review identified solutions. Based on the input from the challenges and the solutions, we created a value stream map of the current and future process.
Conclusions: Overall, we found that the evidence-based process as presented in this study helps in technology transfer of research results to industry, but at the same time some challenges lie ahead (e.g. scoping systematic reviews to focus more on concrete industry problems, and understanding strategies of conducting EBSE with respect to effort and quality of the evidence).

Key words:
Evidence-based software engineering, Process Assessment, Automotive Software Testing

1. Introduction

Evidence-based software engineering consists of the steps 1) Identify the need for information (evidence) and formulate a question, 2) track down the best evidence to answer the question and critically appraise the evidence, 3) critically reflect on the evidence provided with respect to the problem and context that the evidence should help to solve [1]. Overall, the aim is that practitioners who face an issue in their work shall be supported to make good decisions of how to solve the problems by relying on evidence.

In software engineering two approaches for identifying and aggregating evidence (systematic map [2] and systematic review [3]) have received much attention, which is visible in a high number of systematic reviews and maps published that cover a variety of areas. This ranges from very specific and scoped questions (e.g. within-company vs. cross-company cost estimation [4]) to very generic questions (e.g. what we know about software productivity measurement [5] or pair programming [6]). So far no studies exist that used the evidence

*Corresponding author
Email addresses: `abhinaya.kasoju@systemite.se` (Abhinaya Kasoju), `kai.petersen@bth.se`, `kai.petersen@ericsson.com` (Kai Petersen), `mika.mantyla@aalto.fi` (Mika V. Mäntylä)

based process coming from a concrete problem raised in an industrial case and then attempting to provide a solution for that concrete problem/case through the evidence based process. We extend existing work on evidence based software engineering by using the evidence based process for analyzing an industrial automotive software testing process.

In particular, the first contribution of this paper is to demonstrate the use of the evidence-based process in an industrial case providing reflections of researchers using the process to help the company in their improvement efforts. The process is a staged process where subsequent stages use the input of the previous ones, allowing to provide a traceable and holistic picture from practical challenges to recommendations for improvements. The steps are: identify need for information/problems to be solved (through case study); identify solutions and critical appraisal of those (through systematic literature review); critically reflect on the solutions with respect to the problem and mapping them to solve the problem (through value stream mapping); reflection on the EBSE process.

The second contribution is an in-depth understanding of the automotive software testing process with respect to the current situation (strengths and weaknesses) as well as defining a target process based on evidence presented in literature. In other words, it is shown how an improved automotive software testing process based on evidence from literature would look like. There is a need to better understand and address the challenges related to software testing in the automotive domain (see e.g. [7, 8]), and to identify what solutions are available to automotive companies to deal with these challenges, which is of particular importance due to the specific domain profile of automotive software engineering (as presented in Pretchner et al. [9]).

The remainder of the paper is structured as follows: Section 2 presents the related work, elaborating on the characteristics of automotive software engineering. Section 3 presents the staged evidence-based process using a mixed method design, as well as the research questions asked in each step. Sections 4 to 7 present the different stages of the EBSE process proposed in this study, and their outcomes. Section 8 presents the validity threats. Thereafter, Section 9 discusses the results, followed by the conclusions in Section 10.

2. Related Work

The related work focuses on characterizing the automotive software engineering domain and provides a motivation for the study. Details on solutions for automotive software testing in response to the challenges identified are provided through the systematic literature review (see Section 5).

2.1. Automotive Software Engineering - Characterizing the Domain

Pretchner et al. [9] provided a characterization of the automotive domain based on literature and suggest a roadmap for automotive software engineering research. In the following paragraphs, we highlight the combination of characteristics that make automotive software engineering stand out.

Characteristic 1 - Heterogenous subsystems: Many different types of systems (e.g. multimedia, telematics, human interface, body/comfort software, software for safety electronics, power train and chassis control software, and infrastructure software) are part of cars built today. They are highly heterogenous and as a consequence there are no standards, but instead very different methods, processes, and tools for developing automotive systems.

Characteristic 2 - Clearly divided organizational units: Historically, the automotive industry is characterized by vertically organized units being responsible for different parts of the car. The parts then were assembled. Given that software is more complex and need to enable communication between the systems integration becomes a challenge. A general (but in automotive systems amplified) challenge is that suppliers have freedom in how they realize their solutions, given the lack of well established standards. Therefore, there is a strong need for communication between many different stakeholders. Given the high number of stakeholders involved, there are many sources for new requirements as well, which leads to requirements volatility.

Characteristic 3 - Distribution of software: Previously unrelated mechanical functions are now related due to the introduction of software (e.g. driving tasks interact with comfort and infotainment). The distribution requires that different functional units interact through middleware/buses. Furthermore, multiple real-time and operating systems are embedded in a car. This increases the complexity and may lead to unintentional or intentional feature interactions and hence makes quality assurance harder.

Characteristic 4 - Highly configurable systems: Automotive software is highly configurable. Pretchner et al. [9] state examples of a car having more than 80 electronic fittings, which has to be reflected in the software, and they also report of components having 3,488 different component realizations. In addition, configurations over time change and have different life-cycles. Hardware configurations might have longer life-cycles than electronic units and their respective software implementations. This leads to many different versions of software in a car that result in compatibility problems.

Characteristic 5 - Cost pressure and focus on unit-based cost models: The automotive domain is characterized by cost pressure and has a strong focus on unit-based cost models. Optimizing the cost per unit (e.g. by tailoring a software to a specific processor or memory that is restricted by its capacity) leads to problems later, e.g. when porting or extending/maintaining that software.

2.2. Automotive Software Testing

It has been found that little evidence collected from industry exists on how testing processes are performed in the automotive domain and challenges in this context are not evaluated [8, 10]. Furthermore, interaction between test procedures, methods, tools and techniques with test management and version management is left untold [10]. The need to test as early as possible on multiple integration levels under real time constraints put high demands on the test process and procedures being used [10]. The need to quantify the quality assurance value of testing activities in automotive context was identified by Sundmark et al. [8]. They conducted a detailed study on how system testing is performed in connection to a release process in the automotive context and identified several challenges in this regard. Moreover, they observed a need for detailed identification and prioritization of areas with improvement potential. However, there have been no studies with an in-depth focus on strengths and challenges within the whole test process from a process improvement perspective in the automotive software context.

Hence, the related work underlines the need for gaining a rich understanding of challenges in the domain, and explore which solutions are available for these challenges and mapping those to the software testing process.

3. Evidence-Based Software Engineering Process Used in the Case Study

In evidence based software engineering (EBSE) the "best" solution for a practical problem should be selected based on evidence. EBSE consists of the following steps: 1) Identify the need for information (evidence) and formulate a question, 2) track down the "best" evidence to answer the question and critically appraise the evidence, 3) critically reflect on the evidence provided with respect to the problem and context that the evidence should help to solve. In the end (Step 4), the evidence based process (steps 1-3) should be critically evaluated.

In previous studies the steps of EBSE were conducted in isolation, e.g. case studies investigating challenges and issues (e.g. [11, 12]), systematic reviews are conducted to answer a research question (e.g. [5, 6]), and solutions were evaluated (e.g. [13, 14]).

In this research we use a multi-staged EBSE research process where a subsequent stage builds upon the previous ones (see Figure 1). Furthermore, in order to systematically close the gap between the results from step 1 (Identify the need for information (evidence) and formulate a question) and step 2 (track down the "best" evidence to answer the question and critically appraise the evidence) of the evidence based process we used value stream analysis in step 3 (critically reflect on the evidence provided with respect to the problem and context that the evidence should help to solve).

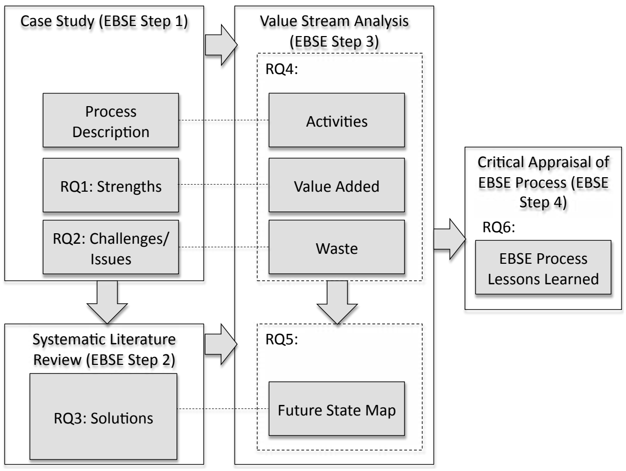
Case Study (EBSE Step 1)
Process Description
RQ1: Strengths
RQ2: Challenges/ Issues
Value Stream Analysis (EBSE Step 3)
RQ4:
Activities
Value Added
Waste
RQ5:
Future State Map
Systematic Literature Review (EBSE Step 2)
RQ3: Solutions
Critical Appraisal of EBSE Process (EBSE Step 4)
RQ6:
EBSE Process Lessons Learned

Figure 1: Staged EBSE Process

The overall goal of the research is to improve the software testing process in the context of automotive software engineering. The stages of the research lead up to the goal as follows:

EBSE Step 1: First, we need to gain an in-depth understanding of challenges and strengths of the testing process to solve the right problems. Case studies are suitable to gain an in-depth understanding of real-world situations and processes [15]. The research questions asked in the case study are:

- *RQ1: What are the practices in testing that can* *be considered as strengths within automotive domain?* An inventory of activities that act as strengths in the testing process is provided through this research question. This is extracted from the qualitative data obtained through interviews.

- *RQ2: What are the challenges/ bottlenecks identified in testing automotive systems?* Lists of challenges or poorly performed practices that act as barriers to incept quality in the testing process are collected to answer this research question.

EBSE Step 2: In the next step we identified the solutions that would help to address the challenges (EBSE Step 1) related to automotive software testing through a domain specific systematic review. We conducted a domain specific systematic review for multiple reasons. First, the automotive domain has specific characteristics, which distinguishes it from other domains. Hence, findings for solutions in the domain context are more likely to be transferable. Second, given that the overall testing process was studied the scope of the review would not be manageable and we would not be able to provide timely input for the solutions. The results of EBSE Step 2 can be seen as an inventory of solutions based on which improvements can be proposed. Results from EBSE Step 1 related to strengths are added to this inventory. The research question asked in the literature review is:

- *RQ3: What improvements for the automotive testing process based on practical experiences were suggested in the literature?*

EBSE Step 3: Based on the detailed definition of strengths and challenges, as well as solutions we used value stream analysis. Value stream analysis was selected as an analytical approach for the following reasons. First, value stream mapping distinguishes between a current state map where the current situation is analyzed with respect to value (what is working well and adds value to the customer) and waste (everything not contribution directly or indirectly to the customer value) and the future state map (desired mapping of the process based on improvements) [14, 16, 17]. The current state map therefore uses the case study as input, while the future state map uses the case study as well as the systematic review in order to map out the desired process representing an evidence-based recommendation to practitioners of how to conduct the testing process. Secondly, value stream mapping has its origin in the automotive domain, which makes its usage in the studied context easy. The following research questions are answered:

- *RQ4: What is value and waste in the process considering process activities, strengths and weaknesses identified in EBSE Step 1?*
- *RQ5: Based on the solutions identified in EBSE Step 2, how should the process represented by the current value stream map be improved?*

EBSE Step 4: In the last step, we reflect on the usage of the evidence-based process in improving software engineering current practices.

- *RQ6: What was working well in using the EBSE process with mixed research methods and how can the process be improved?*

4. EBSE Step 1: Case Study on Strengths and Challenges

We conducted an industrial case study [15] to investigate problems and challenges in test process in automotive software domain and identify improvement potentials, answering RQ1 and RQ2.

4.1. Case Study Design Type

The case being studied is one of the development sites of a large Swedish automotive organization. The case organization is ISO certified. However, the organization was struggling with achieving the SPICE levels their customers desired. In particular, different departments achieved different results in assessments. This is also visible from this study, as we found that there are no unified test processes, and not all projects have proper test planning. They focus on both soft and hard products involving areas such as telematics, logistics, electronics, mechanics, simulation modeling and systems engineering.

We report on a single-case with multiple units of analysis [18], in which we studied the phenomenon of testing in several projects in one company. This type of case study helps comparing between the testing methodologies, methods and tools being used for different projects at the case organization.

The units of analysis here are different projects at the studied company. They were selected in such a way that they have maximum variation in factors such as methodology being used, team size, and techniques used for testing. The motivation for focusing on projects with variation was to be able to elicit a wide array of challenges and strengths. Furthermore, this aids in generalizability as the challenges are not biased toward a specific type of project.

4.2. Units of Analysis

All the projects studied for this research are bespoke as the case organization is the supplier to a specific customer. All the projects here are externally initiated and the organization do not sell any proprietary products/services. Projects within the organization are mostly either maintenance projects or evolution of existing products. It is common within this organization for a role to have multiple responsibilities in more than one project. An overview of the studied projects is given in Table 1.

Systems: The majority of systems are embedded applications (P1, P2, P3, P4, P7, and P8), i.e. they involve software and hardware parts, such as control units, hydraulic parts, and so forth. Windows applications developed in P2, P5, and P6 do not control hardware.

Team size: We distinguish small projects (less than four persons in a team) and large projects (four or more persons in a team). The majority of the teams are large as shown in Table 1. Small teams do not necessarily focus on having a structured development and test process, roles & responsibilities, test methods or tools. Three projects (P3, P6, and P8) did not report any test planning activities. Projects with a higher number of modules are developed by large teams and these projects are old compared to the projects dealt with by small teams. That is, the systems have grown considerably over time.

Development methods: Different software development methodologies are employed within the organization. However model-based development is the prominent one (P4, P5, P7 and P8) and is used with waterfall model concepts. Waterfall means a sequential process involving requirements, design, component development, integration and testing. Agile development using Scrum has been adopted in one project (P2). Small teams involved in maintenance adopted ad-hoc methodologies (P6). Two projects recently introduced some agile practices to incorporate iterative development (P1 and P5).

Tools: Varieties of tools are employed in the projects for testing, such as test case and data generators, test execution tools, defect detection & management tools, debugging tools, requirements traceability and configuration management tools and also tools for modeling and analyzing Electronic Control Units (ECUs). Apart from these tools customized tools are used in some projects when any other tool cannot serve the specific purpose of the project. These tools are usually meant for test execution, which make test environments close to the target environment. Small teams (e.g. P3) do not rely on testing tools, they use spreadsheets instead. Large teams being responsible for several modules use a diversity of tools for organizing and managing test artifacts.

Test levels: As can be seen in Table 1 almost all projects (seven out of eight) had unit testing in place and in five projects Integration testing was used. Unit/basic tests in the projects were similar to smoke tests. However, the unit tests in this context do not have a well defined scope. Half of the projects studied used test automation. However, the evolving test cases were not always updated into automation builds. From the interview data, it was evident that system integration test is not performed by many teams. However, most of the teams assumed integration test can replace system test. As shown in Table 1, other forms of testing, like regression and exploratory testing, were found to be less common and are gaining importance recently within the company.

4.3. Data Collection

The data was collected through interviews and process documentation. However, data from other sources was not collected due to lack of availability and inadequacy with respect to data quality (e.g. quantitative data). The motivation behind using several sources of data (triangulation) is to limit the effects of only one interpretation and by that making the conclusions stronger [19] .

4.3.1. Interviewee Selection

The selection process for interviewees was done using the following steps:

- A complete list of people involved in the testing process irrespective of their role was created.

- We aimed at selecting at least two persons per project, which was not possible from an availability point of view. In particular, for small projects only one person was selected. For larger projects more persons were selected. Furthermore, different roles associated with the testing process should be covered (including developers, managers, and designated testers). However, the final list of employees who participated in the interviews was based on availability in the time period in which the interviews were conducted (March 08.-April 04., 2011).

- We explained to the interviewees why they have been considered for the study through e-mail. The mail also contained the purpose of the study and the invitation for the interview.

Table 1: Overview of Projects (Units of Analysis)

Department	Project	Testing done in project	Methodology	Size	Application Type
Alpha	P1	Basic/unit test (smoke test), system test, integration test, session-based test management, script-based testing, code reviews	Waterfall development with some agile team practices	Large	Embedded System
	P2	Basic/unit test, system test, integration test, regression test, exploratory test	Agile software development using Scrum	Large	Windows application and embedded system
	P3	Basic/unit test, integration test, exploratory test	Waterfall development methodology	Small	Embedded system
Beta	P4	Basic/unit test, script-based testing, automated testing	Waterfall development methodology	Large	Embedded system
	P5	Basic test/unit test, script-based testing, automated testing	Waterfall development with some agile team practices	Small	Windows application
	P6	Integration test, exploratory test	Ad-hoc development	Small	Windows Application
	P7	Basic test/unit test, system test, integration test, regression test, script-based testing, automated test	Waterfall development methodology with model-based development	Large	Embedded system
Gamma	P8	Basic/unit tests, integration tests, exploratory test	Waterfall development methodology with model-based development	Large	Embedded System

The roles selected represented positions that were directly involved with testing related activities or affected by the results of the entire testing process (see Table 2).

Table 2: Description of Roles

Role	Description
Group manager	Responsible for all test resources such as testing tools. Also responsible to see that the test team has the correct competence level.
Test leader	Traditional role responsible for leading all the test activities such as test case design, implementation and reporting defects. Test leader is also responsible for test activities in project and their documentation.
Developer	The developer uses the requirements specifications to design and implement the system. This role is also responsible for all the testing (for some projects only).
Advanced test engineer	Technical Expert that often works with research projects. In order to avoid confusion this role is also termed as developer in the later sections.
Domain expert	As a technical expert this person is responsible for research engineering project who strives to continuously improve testing in their team. In order to avoid confusion this role is also termed as developer in the later sections.
Test/quality coordinator	Responsible to coordinate the all the test activities in the projects and also is responsible for managing the products.
Project manager	Responsible for planning, resource allocation, and development and follow-up related to the project. The requirements inflow is also controlled by this role.

Roles from both the projects and line organization from three departments alpha, beta and gamma (due to confidentiality reasons, the department names are renamed) were included in our study. It is also visible that some roles are related to project work and some are related to line responsibilities within a department, i.e. they support different projects within a department. The number of interviews in relation to departments, projects, and roles is shown in Table 3.

Table 3: Interviewees

Department	ID	Number interviewed	Roles
Alpha	Line	1	Group manager
	P1	1	Test leader
	P2	2	Test leader, developer
	P3	1	Developer
Beta	Line	1	Advanced test engineer
	Line	1	Test coordinator
	P4	3	2 Developers, project manager
	P5	1	Developer
	P6	1	Developer
	P7	1	Domain expert
	P8	1	Developer

In departments Alpha and Beta a sufficient number of employees was available, but in Gamma only one person was interviewed due to lack of availability of persons in that department. The person was selected as she was considered an expert with a vast amount of experience with respect to testing automotive systems.

4.3.2. Interview Design

The interview consisted of four themes; the duration of the interviews was set to approximately one hour each. All interviews were recorded in audio format and also notes were taken. A semi-structured interview strategy [19] has been used in all the interviews. The themes of the interviews were:

1. *Warm up and experience:* Questions regarding the interviewees background, experience and current activities.
2. *Overview of software testing process:* Questions related to test objects, test activities, and information required and produced in order to conduct the tests.
3. *Challenges and strengths in testing process:* This theme captured good practices/strengths as well as challenges/poorly performed practices. The interviewees are supposed to state what kind of practice they used, what its value contribution is and where is it located in the testing process.
4. *Improvement potentials in testing process:* This theme includes questions to collect information about why the challenge must be eliminated and how the test process can be improved.

4.3.3. Process Documentation

Process documentation, such as software development process documents, software test description documents, software test plan documents and test reports have been studied to gain an in-depth understanding of the test activities. Furthermore, documents related to organization and process descriptions for the overall development process have been studied to gain familiarity with respect to the terminology used at the company. This in turn helped in understanding and analyzing the interview data.

4.4. Data Analysis

In order to understand the challenges and strengths in the automotive test process an in-depth analysis of different units of analysis was done using coding. Manual coding was done for 5 interview transcriptions to create an initial set of codes. The codes were clustered into different main categories, predefined by our research question (Level 1), by literature (Level 2) and through open coding (Level 3 and 4), see Table 4. With this a coding guide was developed. For the open coding we coded the transcribed text from the interviews, which evolved. If we, for example, found a new statement that did not fit to an already identified code we created a new code, such as *Interaction and communication*. When we found another statement that falls into an existing code, we linked the statement to that code. After having coded the text, we looked at each cluster identifying very similar statements, and then reformulated them to represent a single challenge/benefit. After having done that we reviewed the clusters and provide a high level description for each cluster. The open coding strategy followed in this research is hence very similar to the one presented in [20]. In order to validate the coding guide, an interview transcription was manually coded by an employee at the case organization and the results of the coding were compared with the researchers' interpretation and required modifications were made. However, the coding guide was continuously refined throughout the data extraction phase.

4.5. Results

The results include a description of the test process, as well as strengths and challenges associated with the process.

4.5.1. Testing Process

The majority of the interviewees (9 interviewees) stated that there is lack of a clear testing process which can be applied to any project lifecycle. Among 8 projects studied only 3 projects have an explicitly defined testing process. It is observed that each project follows a process very similar to what is shown in Figure 2, even though not all projects follow all activities outlined in this process.

A test strategy of an organization describes which type of tests need to be conducted and how they should be used with development projects with minimum risks [24]. The test strategy used at the company was to mainly focus on black-box testing with only a minor part of testing being performed as white-box testing. There is a testers handbook available within the organization which describes test processes, methods and tools. However, this study shows that it is not implemented/used by most of the teams. The main activities conducted are: Test Planning, Test analysis and Design, Test build, Test Execution and Reporting. Among these, test planning is done in advance by five projects (three large teams represented by P1, P2, and P4 and two small teams represented by P5 and P7). Most of the small teams did not have any software test plan even though they had a very flexible test strategy/approach to carry on with tests.

In the following the steps are described in further detail:

Test planning: This activity aims to address what will be tested and why. The entry criteria for this activity is to have prioritized requirements ready for the release as input for test planning. The delivery of this phase is the software test plan, containing estimations and scheduling of resources needed, test artifacts to be created, as well as techniques, tools, and test environments needed. The roles involved in this phase of testing are customer,

Table 4: Analysis through Coding

Coding Level	Descriptions	Purpose
Level 1	Codes directly related to case study research questions i.e., testing practices, strengths and improvement potentials, and problems or challenges are identified here.	Structure statements from interviews according to research questions. Results concerning "testing practices" can be found in Section 4.5.1, results related to "strength and improvement potential" in Section 4.5.2, and results related to "problems or challenges" in Section 4.5.3.
Level 2	Value (five categories) [21], Waste (Seven Categories) (see [22, 23]).	Structure findings according to value (see Table 10) and waste (see Table 12) to be used in the current stream map (see Figure 4). This is then used to map strengths to value (Table 11) and problems/challenges to waste (Table 13).
Level 3	This level defines where in the process practices are implemented.	Identification of process areas (see e.g. Table 5) to clarify the scope of the challenges and being able to map waste to test process activities (Figure 4)
Level 4	Codes derived from interviews (e.g. all aspects related to communication, availability of process documentation, etc.)	Identify groups of related challenges (see C01 to C10 in Section 4.5.3).

project manager and test leader. If there is no test leader available for the project, the developers themselves participate in the test planning activities. The exit criterion for test planning is the approval of the test plan by the customer and project management.

Test analysis and design: This activity of testing aims to determine how the tests (by defining test data, test cases and schedule progress for the process or system under test) will be carried out, which is documented in the software test description. Software test description also defines what tests (i.e., test techniques) will be performed during test execution. The other deliveries during this phase are requirements traceability matrix, test cases and test scripts design to fulfill the test cases. Test cases are written and managed using test case management tools, which are used in all projects. The criterion to enter this phase is to have the software test plan approved by the customer and project management. The test plan scheduled in the previous phase is updated with all detailed schedules for every test activity. The roles involved at this stage are test leader or a test coordinator who is responsible for designing, selecting, prioritizing and reviewing the test cases. Since testers share responsibilities between projects and are not always available for testing tasks, in most of the projects the developers are responsible to write test cases for their own code. The project manager is responsible for the supervision of test activities.

Test build: In automotive software testing, test build is the most vital part of the test process since it involves building a test environment, which depicts the target environment. The outcome of this level is having hardware, which can be visualized as real time environment, including test scripts and all other test data. Since the case organization works with control engines and Electronic Control Units (ECUs) [8] for most of the projects, modeling tools such as Simulink along with MATLAB are used to visualize the target environment. Mostly testers or developers are involved in this activity. The project manager is responsible to provide resources, such as hardware equipment. The test leader supervises the activity.

Test execution and reporting: The final stage of the test process is to execute tests and report the results to the customer. In order to execute tests, the test leader or project manager will chose an appropriate person to run the test scripts. After the tests are completed the results are recorded in the defect management system. The outcome of this phase is a software test report which describes the entire tests carried out as well as their test verdicts. The results are also later analyzed and evaluated to find if there are any differences in comparison to test reports of previous releases. In case of serious errors these errors are corrected and tests are repeated. The project manger is responsible to decide the stopping criteria for test execution.

4.5.2. Strengths and Good Practices

The strengths of the test process are found to be dependent on team size. Most of the practices considered as strengths in small teams were not perceived as strengths in large teams and vice versa. That is, it is evident from the interviews that the strengths vary with team size.

Work in small, agile teams: In small teams test activities are flexible, and there is no need to generate extensive test reports. Large teams do this for small releases only. Large teams have a very structured and plan-driven approach for testing. Small teams focus on continuous integration and iterative development (e.g. P2 using Scrum with continuous integration and sprint planning). Agile test practices make it easier for them to

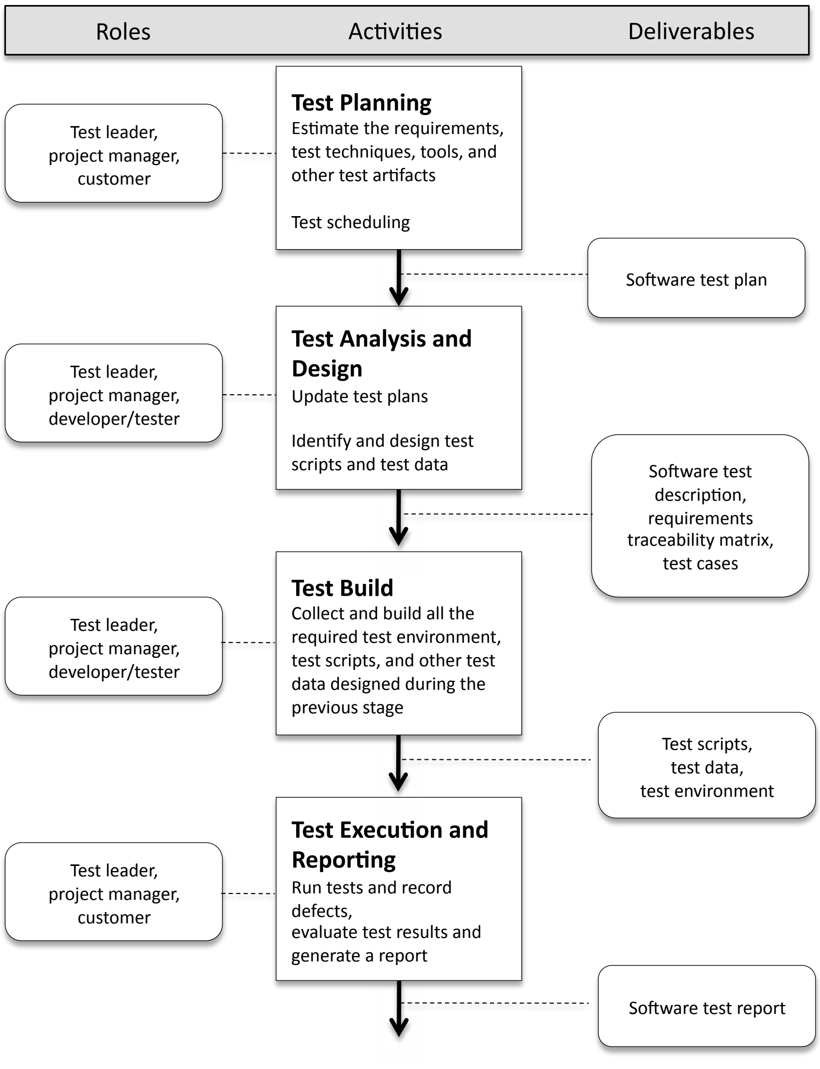
Roles
Activities
Deliverables
Test Planning
Estimate the requirements, test techniques, tools, and other test artifacts
Test scheduling
Test leader, project manager, customer
Software test plan
Test Analysis and Design
Update test plans
Identify and design test scripts and test data
Test leader, project manager, developer/tester
Software test description, requirements traceability matrix, test cases
Test Build
Collect and build all the required test environment, test scripts, and other test data designed during the previous stage
Test leader, project manager, developer/tester
Test scripts, test data, test environment
Test Execution and Reporting
Run tests and record defects, evaluate test results and generate a report
Test leader, project manager, customer
Software test report

Figure 2: Testing Process

plan tests for every iteration that are compatible with the requirement specification. This in turn enables alignment of testing with other activities (such as requirements, design, etc) properly. In comparison to small teams, large teams have a stronger focus on reusing test cases most of the time, which makes them more efficient.

Communication: Strengths regarding communication are found in a project having agile practices such as stand-up meetings, regular stakeholder collaboration and working together in open office space. Every activity involves a test person, which indicates parallel testing effort throughout the whole development lifecycle. In addition to this the agile approach enhanced the team spirit, leading to efficient interactions between team members, and resulting in a cross- functional team. Other projects use weekly meetings and other electronic services, such as email and messaging within a project.

Shared roles and responsibilities: Small teams consider having one person to perform the tester & developer role as strength since this would not delay the process for having to wait for someone to test the software, one developer was stating that: *"While we are working, since the tester is the same person as the developer, there is no delay in reporting it. So if the Developer/Tester finds out the fault he knows where it is introduced, and instead of blaming someone else, the developer becomes more careful while writing the code"*. However, large teams do not consider this as strength; most of these teams do not have any dedicated testers (except one large team which has dedicated testing team).

Test techniques, tools, and environments: Here we made different observations with respect to size of the projects. In small teams fewer testing tools and methods are used to avoid more documentation. These teams generally have less project modules when compared to large teams. In this case the system is well known to the tester/developer (Developing and testing done by one person in small teams) which makes it easy to test using minimum number of tools and methods. Small teams (for example, projects P3 and P6) generally perform smoke or unit test which tests the basic functionality of the system and then have an integration test. An employee conveys the use of unit/basic test in the following way: *"I think unit testing is a strength. With this one goes into details and make sure that each and every subsystem works as it is supposed to"*. Tools for testing used here are developed by the teams to suit the project requirements. However, these customized tools developed for their specific team are not shared among the teams. The main focus in small teams is to have a test environment that has the same hardware and interface as the target environment. This makes maintenance of tests efficient within a project.

Contrary to the small teams, large teams use a variety of methods and tools for testing to perform multiple activities. One of the most perceived strength found in large teams is experience-based testing (e.g. projects such as P1, P2, P4, and P8). As the same team members have been working on the same project over the years, they find it easy to use their experience based knowledge in product development and testing. An employee responsible for quality coordination in a large team says *"The metrics used for testing are not very helpful to us as a team as testing is more based on our experience with which we decide what types of test cases we need to run and all"*. The other perceived strength is exploratory testing/session-based test management applied in projects P1 and P2. An employee pointed out *"Executing charters for session based tests (i.e., exploratory tests) we find critical bugs at a more detailed level"*. Hardware in the loop (HIL) is also considered as strengths for one of the large teams since it detects most of the defects during integration testing. HIL used for integration and system level testing is perceived as a strength as it detects the most critical defects such as timing issues and other real time issues for large and complex systems. Informal code reviews are considered as strength in large teams even though they are also used in small teams. Informal code reviews avoid testing getting biased since it is performed by the person other than the one who is responsible for coding.

Coming to the tools, test case management tools are considered as an advantage in large teams (e.g., P4) as one employee pointed out *"I think test case management tool is a great way to store the test cases and to select the tests that should be performed and also for the tester to provide feedback"*. Other tools considered useful are defect management tools (for e.g., projects). Test environment in large teams is quite good for testing as it depicts real time environment

4.5.3. Challenges

Challenges are grouped into challenge areas. For each challenge area, we also state the number of projects for which the challenges within the challenge area were brought up, as well as the process areas that were concerned by the challenge area (see Table 5), and in each area a set of related issues is reported.

Table 5: Overview of Challenge Areas

ID	Challenge area	No. of projects	Process area
C01	Issues related to organization and its processes	6	Requirements, test process, test management, project management
C02	Time and cost constraint related hinders	5	Requirements, project management, test level (basic/unit test)
C03	Requirements related issues	3	Requirements, test process (test planning), project management
C04	Resource constraints related issues	3	Test process, project management
C05	Knowledge management related issues	5	Test management, project management
C06	Resource constraints related issues	3	Test process, project management
C07	Test techniques/tools/environment issues	2	Test process, test management, test levels
C08	Quality aspect related issues	3	Test process
C09	Defect detection related issues	2	Test process, test management
C10	Documentation related issues	2	Test process

C01: Organization of test and its process related issues: Organizational issues relate to poorly performed practices related to organization and its test processes, such as change management, lack of structured test process, etc. Organizational issues also include stakeholders attitude towards testing (If testing is given low priority).

C01_1: No unified test process: Projects vary in their use of testing methods and tools, and it was considered challenging to find a unified process that suits all projects because of scattered functionality and evolving complexity in hard- and software. Even though a testers handbook is available that might help in achieving a more unified process, it is not used as teams are not aware about it, or people do not feel that it suits their project characteristics. Unstructured and less organized processes work well for the small projects, but not for the larger ones, as it compromises quality. As interviewees pointed out *"It feels there is lack of a structured testing process and it is also un-organized always. It works fine for small projects, but not for large projects"*.

C01_2: Testing is done in haste which is not well planned: The delivery date is not extended when more time would be needed, which results in testing being compromised and done in haste. Furthermore, the customer does not deliver the hardware for testing in time and good quality, hence tests can not be done early; a consequence is a generally low respect for deadlines with respect to testing.

C01_3: Stakeholders attitude towards testing: Improvement work in the past has been focused on implementation, not testing. Hence, new approaches for testing do not get much support from management, which sometimes makes teams develop their own methods and tools, which requires high effort.

C01_4: Asynchronous testing activities: Testing is not synchronized with other activities related to contractors; test artifacts have to be re-structured in order to synchronize with the artifact supplied by the contractor. This leads to rework with respect to testing.

C02: Time and cost constraints for testing: Challenges regarding time and cost constraints can be due to insufficient time spent on requirements, testing activities or test process.

C02_1: Lack of time and budget for specifying validation requirements: Validation requirements are requirements, which are validated during testing (e.g. specifying environmental conditions under which the system has to be tested). Time and money saved on not writing the validation requirements lead to a lot of rework and time in other parts of the process, specifically testing. As one interviewee was pointing out: *"Re-write customer specifications into our own requirements? That is not possible today due to the reason that customer will not pay for it and we do not have internal budget for that'*. Overall, the lack of validation requirements leads to a lack of objectives and a defined scope for testing.

C02_2: Availability of test equipment on time: Test equipment not being available on time and in good quality resulted in unit testing not being conducted.

C03: Requirements related issues: Insufficient requirements for testing, high-level requirements which are hard to understand, and requirements volatility are the challenges that hinder performing proper testing to achieve high quality. These issues generally occur when customer does not specify requirements properly due to lack of time or lack of knowledge which implies poor requirements management.

C03_1: Lack of requirements clarity: Too little effort is dedicated to understanding and documenting clear requirements, resulting in much effort in re-interpreting them in later stages, such as testing. As one of the employees specifies: *"I think it would be better for us in beginning to put greater effort with requirements management to avoid customer complaining about misunderstanding/misinterpreting the requirements specified by them, in order to have fewer issues at the end and save time involved in changing and testing everything repeatedly"*.

C03_2: Criteria for finalizing test design and start/stop testing are unclear: According to the interviews, defining the test process would be completed once the requirements are stable. The interviewees connected requirements volatility to start and stoppage criteria for testing. Requirements volatility required redefining the entire test planning. This acted as a barrier in starting with the actual tests. In cases where the organization used test scripts to perform tests, they had a hard time defining when to stop scripting and start/stop tests as requirements were pouring in. The criteria of when to stop testing were mostly related to budged and time, and not test coverage.

C03_3: Requirements traceability management issues: The traceability between requirements and tests could be better in order to easily determine which test cases need to be updated when the requirements change. Furthermore, a lack of traceability makes it harder to define test coverage. The reason for lacking traceability is that requirements are sometimes too abstract to connect them to concrete functions and their test cases.

C04: Resource constraints for testing: These challenges are related to the availability of skilled testers and their knowledge.

C04_1: Lack of dedicated testers: Not all projects have dedicated testers, instead the developers interpret the requirements, implement the software, and write the tests. The lack of independent verification and validation (different persons write the software and test the software) leads to a bias in testing.

C04_2: Unavailability of personnel for testing: Given the complexity of the systems, building knowledge to be a good tester takes time. In case the experienced testers are shifted between the projects it is hard to find someone who can complete the task at hand. An interviewee who manages testing says *"It is difficult to find people with same experience and also they take quite long period to learn and get to know about the product due to its complexity. For this one need to have same knowledge before being able to do testing"*.

C05: Knowledge management related testing issues: The issues related to knowledge management found in this case studies are:

C05_1: Domain and system knowledge transfer and knowledge sharing issues regarding testing: New testing techniques (exploratory testing) used at the company require vast amount of knowledge, which is not available due to that testers always change and new testers employed by the studied company come into projects. No sufficient information and training material is available on how to test, even though there is a need to achieve a status where a project is not dependent on a single person. From the interviews we also found that the challenge of knowledge transfer is amplified as beyond software it has an emphasis on control engineering, mechatronics, and electrical engineering.

C05_2: Lack of basic testing knowledge: Testing is given low priority due to that testers lack basic testing knowledge. With regard to this context an interviewee involved in life cycle management activity stated that *"I think there is lack of information on testing fundamentals. Some of us do not know when to start a test level and when to end it and it feels like grey areas which is not clearly defined anywhere"*.

C06: Interactions, communications related issues in testing: Problems in practices related to communication between different stakeholders in involved in testing. Also includes improper form of communication such as lack of regular face-to face meeting, lack communication between customer and test personnel.

C06_1: Lack of regular interactions with customer regarding requirements: In the beginning of projects customer interaction is more frequent, with respect to validation requirements in testing there is too little customer interaction. The right person to communicate with regarding requirements on testing is unavailable on the customer side.

C06_2: Lack of interactions with other roles within project during testing: There is a lack of communication with previous team members that have shifted to another project, even though they are needed (e.g. in order to verify and fix identified bugs). One interviewee narrated it in the following way; *"I have allocated a person for our team and then he have to communicate with us but it has been sometimes quite tough for the person to find the person since he is working for another team now"*.

C06_3: Informal communication with customer: Overall, there is a lack of face-to-face and informal communication with the customer and the customer communicates by providing vague descriptions, which are then not clarified. A manager adds *"I think it is most critical to maintain the relationship (informal relationship with customer) and demand the customer that we cannot start working before you tell us what you want"*.

C07: Testing techniques, tools and environment related issues: Problems related to usage of current test techniques, environment and tools.

C07_1: Lack of automation leading to rework: The automation of unit tests and regression testing is not done efficiently. One interviewee pointed out that *"Testing is rework as long as it is not properly automated"*. Generating and efficiently automating tests is observed as a challenge due to a perceived lack of unavailability of tool support, leading to rework when wanting to rerun tests.

C07_2: No unified tool for entire testing activity: One test lead pointed out the need for unified tool which can be used for testing *"we have lot of tools for testing but there are some difficulties in deciding which tool to use since there are drawbacks and strengths for every tool being used. Sometime we are forced to develop customized tool because we cannot get any tool from the market that does everything for us"*. A tool which does all activities in testing for automotive domain can be easy to use instead of managing and organizing large number of tools used right now.

C07_3: Improper maintenance of test equipment: Several test environments are to be maintained, lack of maintenance leads to rework and long lead-time before actual testing can start. One interviewee nicely summarized this as *"We have several test environments and test steps to be maintained. They are not always maintained and it takes long time before one can get started with actual testing"*.

C08: Quality aspects related issues: Problems related to incorporating quality attributes of testing such as reliability, maintainability, correctness, efficiency, effectiveness, testability, flexibility, reusability, etc. Involves tradeoffs between quality and other activities.

C08_1: Reliability issues: Reliability of the system is not achieved to the degree desired. Quality is hard to incorporate due to lack of test processes, and due to faulty hardware components. As one interviewee specifies *"Its hard to achieve several requirement criteria for a system such as working for longer period of time, less resource intensive, ability to work on different platforms, etc."*.

C08_2: Quality attributes are not specified well right from the inception of project: Quality requirements are not well specified, leading to a situation where complex systems had quality issues on the market for existing products.

C08_3: No quality measurement/assessment: Quality measures are not present, but their need is recognized to increase the ability to evaluate the results of testing, one employee saying that: *"the quality curve must be better although our customer is satisfied. I think the quality measures should be documented in order to facilitate better analysis of test results"*.

C09: Defect detection issues: Problems related to practices which disable the tester to trace the defect or the root cause of defect creation, also includes problems related defect prevention.

C09_1: Testing late in the process makes it costly to fix defects: Due to the system complexity and late testing the number of defects in the system increases while it evolves and increases in size. Missing many defects in previous releases led to a high number of customer reported defects in the following releases that needed to be corrected, which made defect fixing costly.

C09_2: Hard to track defects which are not fixed in the previous releases: For development with complex parts (i.e., which involves working with timing issues and other critical issues) the difference in the behavior of the system between two different releases need to be same. But this is not always happening due to the errors which were not fixed during the previous releases being triggered in the current release. This is because these errors may become serious in the next releases when they become untraceable in such a huge system.

C10: Documentation related issues: Poorly performed practices related to test documentation such as insufficient documentation, no documentation or too much documentation that does not give proper support for maintaining quality in test process are subject of this challenge area.

C10_1: Documentation regarding test artifacts is not updated continuously: The interviewees emphasized that the documentation (such as test cases and other test artifacts) provided was not enough for testing and cannot be trusted; one interviewee added that *"The test documents are not updated continuously, so we find them unreliable"*. One of the reasons mentioned was there were small changes being done to the test artifacts, which are not which are not updated accordingly the test document. Not updating documentation led to rework.

C10_2: No detailed manuals available for some specific test methods and tools: Another observation in this regard was a lack of documentation on how tools and methods work which can be used. One interviewee nicely summarized this as *"There is support for tools, but we always cant find someone who can fix the problems with them. It could be better documented I guess"*. However, it is observed that there are manuals within the organization which serve this purpose. But for some specific tools (such as customized tools) or methods, this does not work. This issue seems to arise when people performing testing could not understand the terminology in manuals or they are not aware of these manuals.

5. EBSE Step 2: Identifying Improvements through Systematic Literature Review

This section describes the purpose and design of our method in systematic literature review and value stream mapping to improve the testing process. The research question for this part is *RQ3: What improvements for the automotive testing process based on practical experiences were suggested in the literature?*

In Section 4 we identified the challenges related to software testing in the industry. In this section, we present EBSE Step 2. We perform a domain specific systematic literature review to study the state of the art. Second, we create solutions proposals based on the results of the review to address the challenges identified. Doing this is with the spirit of evidence-based software engineering where one needs to consult the evidence base when creating diagnosis and solution proposals [1].

5.1. Systematic Literature Review

The purpose of our SLR is to identify testing related problems in the context of the automotive software domain and solutions that have been proposed and applied in an industrial context. Our SLR design consists of several steps that are presented below. Our SLR is based on guidelines provided by Kitchenhamn [25] with the exception that we did not exclude studies based on quality, as the goal was to identify all potential solutions that are based on industry experience, and not discarding them due to e.g. lack of reported procedures.

The steps for our literature are:

- Define research question for the review.
- Identification of papers
- Study selection
- Results of mapping of solutions to identified challenges

5.1.1. Identification of papers

In this step we formulated search terms so that they enable the identification of research papers. Search terms were elaborated over several test searches in digital libraries. To this end we used five different search strings (see Table 6). The first two strings identify articles on testing in the automotive domain, and model-based tools to support automotive software development in order to cover solutions for challenge areas related to testing. Requirements issues identified were very general requirements problems, but had an impact on testing. Hence, these are also covered in a separate search string. Given that some projects were working in an agile way of working, which was deemed a strength, we also looked for studies related to agile in automotive.

Table 6: Search Strings

Search	Search string
SLR_1	automotive AND software AND (test OR verification OR validation)
SLR_2	automotive AND software AND model-based AND tool
SLR_3	automotive AND software AND requirements
SLR_4	automotive AND software AND (agile OR scrum OR extreme programming OR lean)
SLR_5	embedded AND software AND (agile OR scrum OR extreme programming OR lean)

Search string were applied on Titles and Abstracts in the databases IEEEXplore, ACM Digital Library, Springerlink, ScienceDirect and Wiley Interscience. We did not apply search string on full text as it is found that such approach generally yields too many irrelevant results.

5.1.2. Study Selection

To select papers relevant to our goal we formulated inclusion/exclusion criteria. First of all, we excluded papers that are not in English, published before 2000 (given that in recent years cars contain a vast amount of software and challenges are more related to recent research) and were not available in full-text. As our goal was to look for problems and solutions offered in peer-reviewed literature, we excluded editorial notes, comments, and reviews and so on. As we intended to look for solutions that were applied in industry, we included papers with solutions that have empirical evaluations in industry and in particular automotive software domain. A major criterion to include a study was that they present solutions to problems in relation to software testing. By software testing, we mean any of the V&V activities spanning across the whole software development lifecycle (requirements validation, test case generation, unit or regression testing and so on). To ensure these criteria are satisfied, papers were scanned against the checklist.

- Is the paper in English?
- Is the paper available in full text?
- Is the paper published in or after 2000?
- Is the context of research automotive software domain?
- Does the paper talk about any problems and solutions or tools related to any software V&V?
- Does the paper contain an empirical evaluation in industrial context?

The search for SLR_1 and SLR_2 resulted in 221 papers for SLR_1 and 66 papers for SLR_2. An overview of the distribution of primary studies across databases is shown in Table 7, also showing the number of finally selected studies.

The search for SLR_3, SLR_3, and SLR_4 resulted in 301, 12, and 107 papers, respectively. An overview is given in Table 8.

5.1.3. Solutions Based on Systematic Literature Review

We mapped the identified challenges and solutions offered in our SLRs to the challenges found in our interviews. Furthermore, we state other references where the challenges observed in this study have been found. These are shown in Table 9. Based on our SLRs, we present seven solution proposals. It is important to point out that there often cannot be a single solution proposal for each issue. It entirely depends on the type of projects and appropriate strategies (such a resource management, budget management) adopted by teams to implement these solutions.

The number of references in relation to the solution proposals was determined by the availability of information. When we created the categories we aimed at not having a category that requires a small/easy solution and another category that covers a large area of solutions. Even though test management (SP7) has only one reference, we believe that research focus on test management is as large in scope as, for example, test automation and tools, looking at its complexity and how hard it would be to solve it. Overall, scope of the problems was the base to decide on the granularity of categories.

SP1: Requirements management (RM): Overall, we identified that good requirements are a prerequisite for good testing in automotive software development. Requirements related issues such as Lack of

Table 7: Number of selected studies (SLR_1, SLR_2)

Database	Initial search result		Nr. Primary studies		Full text not available	
Searches	SLR_1	SLR_2	SLR_1	SLR_2	SLR_1	SLR_2
ScienceDirect	35	4	5	-	-	-
ACM Digital Library	12		4	-	-	-
WileyIntercience	5	4	-	-	-	-
Springerlink	46	16	8	4	13	-
IEEE Xplore	123	23	12	1	-	-
Total	221	66	**29**	**5**		

Table 8: Number of selected studies (SLR_3, SLR_4, SLR_5)

Database	SLR_3		SLR_4		SLR_5	
	Total	Selected	Total	Selected	Total	Selected
IEEEXplore	163	10	5	-	37	3
ACM Digital Library	102	-	1	-	36	-
SpringerLink			3	-	3	-
ScienceDirect	31	-	3	-	17	-
Wiley Interscience	5	-	0	-	14	1
Total	301	10	12	0	107	4

requirements clarity (C03_1) , Requirements volatility C03_2), Requirements traceability (C03_3) can be tackled through better requirements management. Furthermore, we can understand that Quality attribute specification problems (C08_2) as well as customer communication problems (C06_1, and C06_3) can be improved with ideas from the requirements engineering domain. Our domain specific SLR was able to find many solutions to these problems (see Table 9). For example, Grimm from DaimlerChrysler[1] recommends early simulations of requirements and derivation of test cases from specification and suggests tracing and administering requirements across the entire software development lifecycle [7]; Islam and Omasreiter [29] presented and evaluated an approach, where text-based use cases are elicited in interviews from various stakeholders to elicit and specify user requirements for automotive software; and Bühne et al. [30] proposed abstraction levels of Software, Function, System, and Vehicle. Each requirement on each abstraction level is in turn linked to system goals and scenarios.

SP2: Competence management (CM): Competence management was identified to address a number of challenges. We identified a need for Competence Management based on the issues of Lack of dedicated testers (C04_1), Unavailability of personnel for testing (C04_2), Knowledge transfer (C05_1) and Lack of fundamental testing knowledge (C05_2). Our domain specific SLR was able to find only one source by Puschnig and Kolgari [32] who propose means for the involvement of experts in sharing knowledge and expertise with less experienced testers in projects, e.g. workshops are recommended where test team communicate with experts to communicate information on testing, tools as a means for informal training.

SP3: Quality assurance and standards: Quality Assurance and Standards can also help with several problems. Cha and Lim [55] propose a Waterfall type process for the automotive software engineering where quality assurance is performed with peer reviews on artefacts produced in design, implementation and testing phases. Towards this end, DaimlerChrysler developed their own Software quality management handbook for their automotive projects [27], helping to address C01_1/C01_2 which are related to lack of unified test process definitions and test planning. A Draft of ISO 26262 (Road vehicles Functional safety) defines the artefacts and activities for requirements specification, architectural design, implementation and testing, system integration and verification. The standard also prescribes the use of formal methods for requirements verification, notations for design and control flow analysis, the use of test case generation and in-the-loop verification mechanisms, e.g. hardware in the loop, software in the loop, [52] (related to challenge C07_3 on test equipment). Controller Style Guidelines For Production In-

[1]Now Daimler AG after selling the Chrysler Group in 2007

Table 9: Mapping of Challenge Areas to References of Solutions

Nr.	ID	Challenge	Sources	Process Areas
1	C01_1	No unified test process/approach	[26, 27]	Test Management
2	C01_2	Testing done in haste and not well planned	[28, 29]	Agile
3	C01_3	Stakeholders attitude towards testing: low priority	[28]	Agile
4	C01_4	Asynchronous test activities	[28]	Planning/Process
5	C02_1	No time and budget allocated for specifying validation requirements	-	Planning/Process
6	C02_2	Unavailability of test equipment on time	[28]	Test Management
7	C03_1	Lack of requirements clarity	[7, 30, 31, 32, 27, 33, 29, 34, 35]	Requirements Management
8	C03_2	Criteria for finalizing test design and start/stop testing are unclear	[7, 30, 36]	Requirements Management
9	C03_3	Requirements traceability	[37, 7, 33, 38, 30, 27, 36, 34]	Requirements Management
10	C04_1	Lack of dedicated testers	[32]	Competence Management
11	C04_2	Unavailability of personnel for testing	[32]	Competence Management
12	C05_1	Knowledge transfer and sharing issues regarding testing	-	Competence Management
13	C05_2	Lack of testing fundamentals	-	Competence Management
14	C06_1	Lack of regular interactions with customer regarding requirements	-	Requirements/Agile
15	C06_2	Lack of interactions with other roles within the project during testing	-	Test Management
16	C06_3	Informal communication with the customer	-	Requirements
17	C07_1	Lack of automation for test case generation leading to rework	[39, 40, 41, 7, 42, 43, 44, 45, 46, 10, 47, 48, 49, 50, 28]	Automation
18	C07_2	No unified tool for entire testing activity	[7]	Automation/Tool
19	C07_3	Improper maintenance of test equipment	[51, 52]	Test Management
20	C08_1	Reliability issues	[39, 53, 50]	Automation
21	C08_2	Quality attributes are not specified well	[51, 27, 54]	Requirements
22	C08_3	Lack of quality measurement/assessment	-	Quality Assurance
23	C09_1	Testing late in the process makes it costly to fix defects	-	Agile/Defect Management
24	C09_2	Hard to track defects which are not fixed in previous releases	-	Agile/Defect Management
25	C10_1	Documentation regarding test artifacts is not updated continuously	-	Agile
26	C10_2	No manuals for test methods and tools	-	Test Management

tent Using MATLAB, Simulink and Stateglow is a modeling catalogue for Simulink models in the context of automotive systems developed by The MathWorks Automotive Advisory Board (MAAB) [56]. MAAB is an association of leading automotive manufacturers such as Ford, Toyota and DaimlerChrysler.

SP4: Test automation and SP5: test tool deployment: Automation is clearly one of the most important issues in industry and there is a considerable amount of research describing the state of the practice (C07_1). As can be seen in Table 9, numerous test automation solutions have been proposed and used in the automotive domain. For example model-based black box testing [39] [40] is proposed for systems that have high safety and reliability requirements (C08_1); evolutionary testing has been proposed in many works [41] [43] [57] as a solution with the challenge of automating functional testing, and it has been successfully implemented at DaimlerChrysler [44] with a tool called AUSTIN [48] (C07_2); Furthermore, other type of testing and quality assurance tools have been proposed such as Classification-Tree editor CTE [58] for a systematic approach to the design of functional test cases [7]; semi-automatic safety and reliability analysis of the software design process [59] that has been validated in a case study by Volvo involving 52 active safety functions [60]; and a tool for resource usage and timing analysis [61] (C08_1). Overall it can be concluded that when it comes to test automation or tools there is no shortage of proposals focusing on the automotive domain.

SP6: Agile incorporation: In recent years Agile development methods have become popular in the industry and it can also help with many problems experienced in the case company. Based on our interviews, we identified a need for change in the software development process used in the case organization to cope with requirements changes (C02_1, C03_1, C03_2), and here agile process are a natural fit as they offer regular communication where requirements can be changed or clarified. Agile also emphasizes collaboration and communication that can be seen as a solution to knowledge transfer and interaction issues identified (C05_1, C06_1, C06_2, C06_3). Finally, some agile methods emphasize continuous and automated testing, which can potentially help with many testing problems experienced (C01_1, C01_2, C09_1 and C09_2). Although agile can be theoretically linked to many problems our domain specific SLR was not able to find many academic publications of agile. Agile development has been implemented at DaimlerChrysler [62] and it was observed that Agile offers flexibility, high-speed development and high quality. Mueller and Borzuchowski [28] report experiences in using Extreme Programming (XP) on an embedded legacy product and they report that TDD and automation of Unit tests were the essential ingredients for success.

SP7: Test management: From the above identified solution proposals it can found that most of the activities are concerned with the organization of testing and its artifacts. It was also evident from our interviews that most of the challenges identified in the case study were more or less related to management of test activities. There was no study identified that necessarily concentrates on test management activities related to automotive domain. However, very few articles were found in literature which describes the activities that test management must concentrate in order to improve testing which suits this study context. So this solution proposal was formulated to coordinate the above proposed solutions. Test management [63] activities as observed in our study can incorporate the following activities.

- *Test Process management:* Manages various activities within test process such as test planning, test analysis, test build and test execution. This activity is also applicable when agile practices are introduced.

- *Test artifacts and assets organization:* Reuse and maintenance of test artifacts such as test cases, test versions, test tools, test environment, test results and test documentation. This activity can also be termed as test configuration management with which change throughout the life cycle of test activities can be managed.

- *Requirements management in accordance to testing:* Responsible to analyze and determine requirements change which facilitate reasonable adjustment in test schedule and test strategy and thus improve test cases to fulfill new requirements.

- *Competence management:* Responsible to allocate test personnel with required stock of skills and knowledge necessary to perform the specific testing activity.

- *Defect management:* Responsible for early detection of defects that need to be effectively managed and supported through various stages by different people working together.

6. Step 3: Critically Reflect on the Evidence and How to Use it to Solve Problems in the Current Situation

Value stream mapping (VSM) as a process analysis tool is used to evaluate the findings of strengths and weaknesses. This tool is used for uncovering and eliminating waste [23, 14]. A value stream captures all activities (both value added and non-value added) currently required to bring a product through the main process steps to the customer (end-to-end flow of the process). Value adding activities are those that add value to a product (e.g. by assuring the quality of a feature), while non-value added refers to waiting time. The biggest delays or bottlenecks (i.e. non-value added) in a value stream provide the biggest opportunity for improving the process capability [23]. The motivation behind choosing VSM is because it is an efficient tool, with which we could walk through the testing process to understand workflow and focus explicitly on identifying waste with an end-to-end perspective [16]. It provides managers the ability to step back and rethink the entire process from a value creation perspective [14]. Furthermore, it comes natural for the automotive industry and is easily accepted as an improvement approach there, as it originates from the automotive domain (see e.g. Toyota Product Development System [22]).

A value stream map is done in two steps. In the first step the current activities are mapped using the notation in Figure 3, distinguishing value adding and non-value adding activities. Through burst signals wastes and inefficiencies are indicated. Seven wastes are commonly defined for software engineering (see Table 12) [22, 23]. Thereafter, a future state map is drawn which incorporates improvements to the identified wastes. Figure 1 shows how the information obtained from the test process assessment done in the case study maps to the value stream activities.

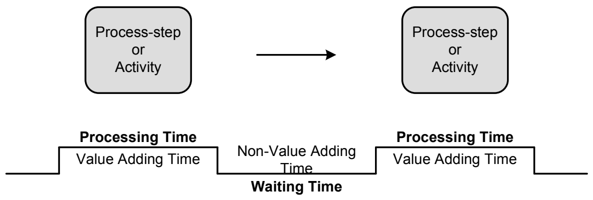

Figure 3: Value Stream Mapping Notation

6.1. Current State Map

We performed a process activity mapping with which we visualized various activities carried out within test process. This section presents the current value stream map, which provides an overview of wastes identified in VSM and the interviews. The values created by the process are identified for various team sizes which are presented in Table 10 (definition of value) and Table 11 (overview of values in the process).

Table 10: Value Definition

ID	Value	Description
V01	Functionality	The capability of the tested product/ service to provide functions which meet stated and implied needs when the software is used under specific conditions.
V02	Quality	The capability of the software delivered after testing to provide reliability, usability and other test attributes.
V03	Internal Efficiency	Represents proper integration of both products features tested for and test process deployment for better organization of critical complexities within the testing activity with respect to time, cost and quality.
V04	Process Value	Quality of entire test process in installing/upgrading/receiving the tested artefact with respect to time, cost and quality.
V05	Human Capital Value	Refers to the stock of skills and knowledge embodied in the ability to perform labour so as to produce economic value with the testing being done.

The non-value adding activities are identified in the current value stream of test process as shown in Figure 4 in order to see where improvements are needed.

The current state map of the test process revealed all seven kinds of wastes as they are defined in [23] in the context of lean software development/value stream mapping. The seven kinds of wastes identified are partially done work, extra processing, handoffs, task switching, relearning, delays and defects (numbered as W1-W7) (see Table 12 for waste definitions).

These wastes are identified in different activities within test process which cause rework, increase in waiting times or inefficient time spent within the entire test activity. Figure 4 illustrates the mapped out test process and the wastes identified. However, the issues that occur in other activities (e.g., requirements management, etc) which affect testing are not shown in the current stream map. The reason behind their cause and their negative influence on test activities are discussed in the previous section.

We identified twelve areas (1-12 as shown in Figure 4) in the test process where wastes occur. Below is a description of the wastes that occur in every sub-process as identified in the current stream map.

Waste identified in sub-process 1: The waste ob-

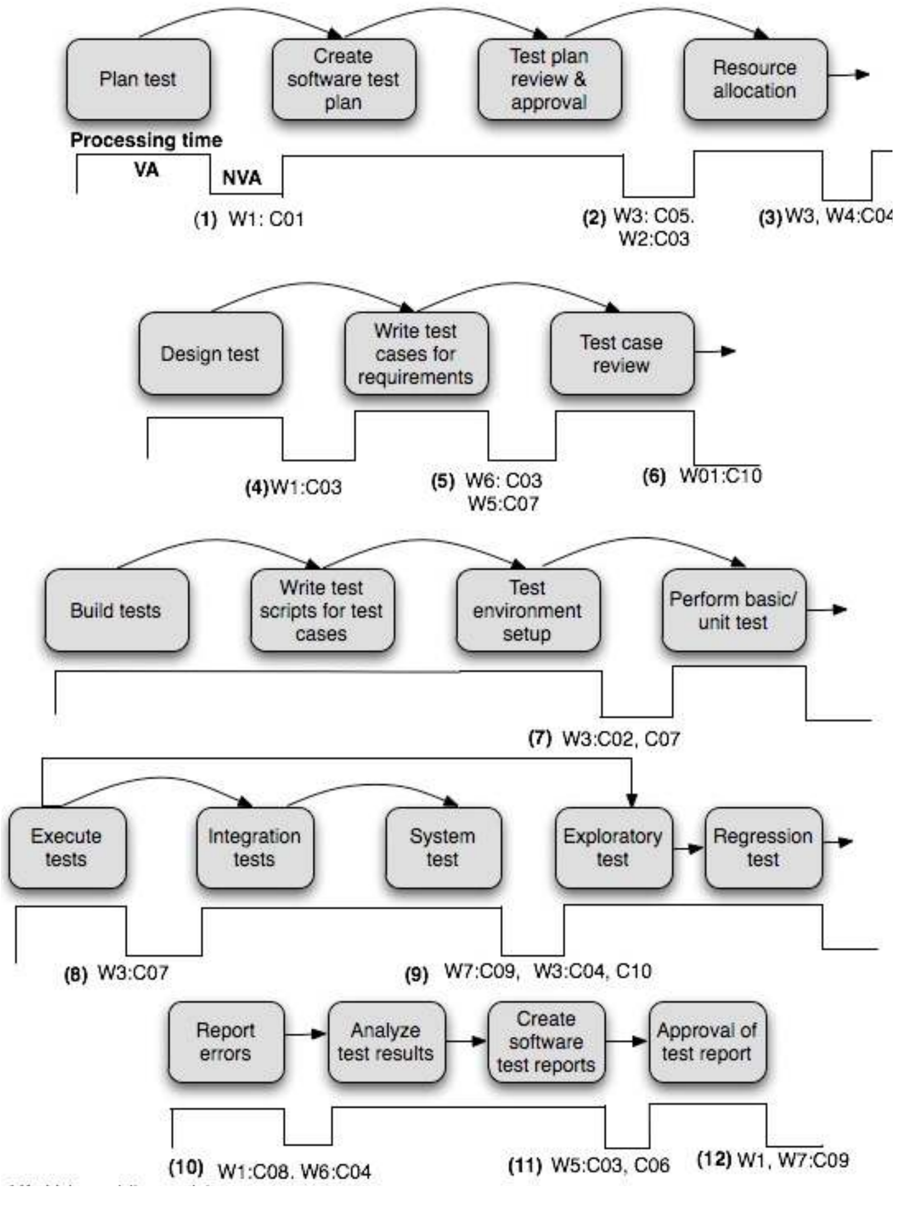

Figure 4: Current State Map

Table 11: Value

Strength	Small team	Large team	Description	Value added
Less documentation	√		More time is spent on delivering functionality	V01
Basic/unit test	√	√	Enables to deliver quality functionality	V01
Integration test		√	Incorporates efficient way of testing by detecting more defects in less time	V01, V03
Test environment depicts target environment	√	√	Better deployment of test process with test environment	V03
Experience based testing		√	Incorporates quality aided by tester's skill and knowledge	V02
Exploratory testing/session-based test management		√	Compatible to the project requirements and aids defect prevention.	V02, V03
Testing tools		√	Better organization of test activities.	V03
Continuous integration	√		More functionality.	V01
Iterative development and testing	√	√	Better functionality and quality.	V01, V02
Roles and responsibilities	√		Flexible roles and responsibilities refers that various stock of skills and knowledge used for testing.	V05
Verification activities (information code reviews)		√	Quality incorporation.	V02
Reuse		√	Test artefacts from previous releases are organized and reused.	V03

Table 12: Waste Definition

ID	Waste	Description
W01	Partially done work	Test activities which are not completely done such as unfixed defects, undocumented test artefacts or not testing at all.
W02	Extra features	Testing features/functionalities that are not required by the customer.
W03	Relearning	Misinterpretation caused due to no documentation of any activity that negatively affect testing. Ex: misinterpreted requirements.
W04	Handoffs	Lack of availability, knowledge or training in adopting compatible test techniques, data, tools or environment.
W05	Task switching	Unclear roles and responsibilities as a part of organization structure with respect to testing which doesnt result in forming right teams.
W06	Delays	Delays that occur to elicit clear validation requirements, approvals and other resources to perform test activities.
W06	Defects	Testing in the end, no early defect detection or prevention activities and lack of verification activities such as code reviews, inspections.

served in sub-process 1 is partially done work. The reason for not completing tests are the lack of planning tests due to lack of test definition and testing done in haste (C1), ultimately resulting in conducting tests in an unstructured manner with low test coverage. This is amplified by unclear requirements.

Wastes identified in sub-process 2: In this process we identified the wastes "extra features" and "handoffs". Extra features that are at times removed from the system prior to release, even though they were implemented, e.g. due to volatile and unclear requirements (C03). However, testing is also performed on such features/functions. This waste occurs in the form of effort that is put in writing the test plan and subsequently scheduling tests and allocating resources. Unclear requirements further require relearning (W3).

Waste identified in sub-process 3: As identified in the case study, one general issue in the case organization is resource constraints (C04). The wastes that occur here are lack of availability of testers (W3: Handoffs) and unclear roles and responsibilities as a part of organization structure ,which hinders the formation of right teams, resulting in task switching (W4).

Waste identified in sub-process 4: Work is not moving forward and gets delayed (W1: partially done work/W6) due to that customer and development organization require much time to negotiate candidate requirements for the current release. It is observed that this process repeats itself numerous times involving several interactions with the customer since no one has the same view as others on the requirements (C03). In order to write test cases for the requirements, there must be a stable and detailed set of requirements to design and analyze the tests for the next release.

Waste identified in sub-process 5: The delay here again occurs in form of long waiting times (W06: Delays) for eliciting validation requirements (C03) to finalize a checklist of test cases to be performed in the test activity. The test cases from the previous releases are sometimes not updated. This takes away lot of time and effort to be spent on rewriting (W5: relearning) the requirements of the previous version and including those test cases in the current release. Lack of automation in test case generation is also a reason for this delay as testing is rework as long as it is not automated (related to testing tools, C07).

Waste identified in sub-process 6: Documentation regarding testing is not always maintained as discussed in challenge C10 earlier. The test cases from the previous release are not always updated to the test case repository which means undocumented test artifacts (W1: Partially done work). Some of these missing test artifacts can put the testing activity into a critical situation, which ends in repeating the entire testing again.

Waste in sub-process area 7: Some projects need test equipment to perform testing. The test equipment from the customer is not available for tests on time (W3: Handoffs). However, this waste is reduced in some cases where the test environment used in the previous releases is saved and maintained for the later versions of the product. As identified in challenge C7, there is no specific reason for this negligence.

Waste in sub-process area 8: All the test activities carried out in the case organization are managed using different tools, which are usually meant to save time. But in practice these tools do not serve this purpose. Instead management and mapping of test artifacts using these tools consume more resources and sometimes redundancy creating unnecessary complexity. A unified tool which can manage and organize all the test activities for automotive domain is not available which makes it a challenge (C7) and thus creating a waste called handoffs (W3), which is related to availability of people, equipment, etc.

Waste in sub-process area 9: Testing is not done as a parallel activity with development (C09). Tracking defects in the end consumes time and money which appears to be a burden on testers leading to huge delays (W6). Verification activities, which support early defect detection, such as inspections and code reviews are not used by most of teams. Another kind of waste (W4: Handoffs) that occurs here can be due to a lack of availability of testers and training for implementing tests using specific testing techniques, such as exploratory tests or experience based testing. Exploratory and experience based testing are based on testers' intuition and skills (see C04). Even though such testing techniques are considered a strength within the case organization, only a limited number of test personnel who have the competence to perform such activities are available right now. This in turn leads to delays in the testing when such experienced testers quit or are shifted to another project. However, documentation on how to use testing techniques and tools are not updated continuously (sometimes not available), and hence cannot be trusted to perform testing (C10).

Waste in sub-process area 10: The quality attributes that need to be incorporated in the tested artifact are not properly elicited since the inception of the project (W1: Partially done work), which leads to poor quality product. Some of the interviewees feel that the testing is being done to ensure quality of basic functionality only, and thus one cannot ensure the reliability of the delivered system (C08). There is a lack of quality standard that is essential to measure the level of quality and to be able to compare test results with previous release. The analysis of test results helps to redefine the quality improvements that need to be implemented in the next versions of the product. Some employees also reported long delays (W6: Delays) for having to wait for the developers to fix the defects after they are reported. This waiting time seems to be long when persons responsible for the code are shifted to other projects as soon as they finished their work in the previous project (see C04). This could be solved if the testing is performed parallel to development.

Waste in sub-process area 11: Due to requirements volatility (C3), the requirements specifications are not documented well, which leads to misinterpretations of requirements. Effort and resources put in developing and testing misinterpreted requirements is not useful (W3: Relearning). Then after a series of interactions with customer the necessary requirements are elicited and developed, which leads to unnecessary rework and task switching (W5).

Waste in sub-process area 12: The defects detected in previous releases are sometimes not fixed (W1: partially done work), which is agreed by the customer. But these defects are difficult to track in the next releases as the system evolves. Lack of verification activities and early defect prevention activities (W7: Defects) creates a mess before release, with which some of the unfixed defects in the current release are left for the next release. This process repeats itself many times during each release. As the functionality grows there are many unfixed defects left behind, which are hard to trace in such complex systems.

A summary of wastes and their relation to challenges is provided in Table 13.

6.2. Future State Map

It is apparent from the results that other processes, especially requirements gathering and documentation, impact testing in a negative manner and led to many wastes. We found that most commonly perceived wastes i.e., W3: handoffs and W1: partially done work were occurring due to long delays in eliciting clear and

Table 13: Waste

ID	Challenges	Description
W01: Partially done work	C01, C03, C08, C09, C10	This waste occurs due to partially done work in terms of test activities such as test plan, requirements, quality incorporation, defect detection and prevention and test documentation.
W02: Extra features	C03	This waste occurs due to extra features developed due to lack of requirements clarity, which are otherwise misinterpreted.
W03: Relearning	C02, C03, C04, C05, C07, C10	This waste occurs to lack of availability of test equipment on time, requirements for testing, and test personnel with required competence in testing, knowledge transfer and knowledge sharing within testing, proper documentation on usage of test techniques and tools. This waste also occurs when there is no test maintenance activity to save test artefacts.
W04: Handoffs	C04	This waste occurs due to lack of dedicated testing team or test personnel, which is due to unclear roles and responsibilities within the team.
W05: Task switching	C03, C06, C07	This waste occurs due to rework caused by misinterpreted requirements.
W06: Delays	C03, C04	Delays in the test process to elicit requirements and allocate resources to perform testing.
W07: Defects	C09	The waste related to defects occur when there are no early defect detection and defect prevention activities, which indicates that testing is done in the end.

stable requirements for testing. The identified challenges in the test process report that continuous inflow of requirements led to reduction in test coverage and increase in the amount of faults due to late testing. The faults that arose in the current release are sometimes not fixed and delivered, due to which the same faults repeat in the next releases, but becomes hard and costly to trace and fix. Hence the testing approach currently used does not suit the continuous flow of requirements, indicating the necessity of shifting to new approach, which can manage and organize changes, and at the same add quality.

The future state VSM is shown in Figure 5 and is agile in nature. The process shown represents one iteration.

We recommend the use of agile practices (SP6) and test management (SP7), which helps to utilize the time of testers more efficiently through parallelization of development and testing, incepting early fault detection, and short ways of communication. Agile can also help in achieving high transparency in terms of requirements for testers since the test planning is done for all iterations. However, test plans can be updated in detail for every iteration. In particular agile practices (SP6) emphasize a requirements backlog and the estimation of resources for iterations to keep them accurate and flexible. At the same time there is a need to document the test plan, as this is a pre-requisite to be able to efficiently reuse test artifacts, and to align testing with requirements activities (proposed in SP7, [63]) for each iteration. To elicit requirements user stories were found useful (see SP1, [29]). Abstraction levels might be of importance as when prioritizing requirements on one abstraction level, the prioritization has to be propagated to the other levels (see SP1,[30]).

A flexible test process is found to be a strength in the projects, especially in small teams. Most of time testing is done in a way that more functionality is delivered (Value: V1) rather than quality. However, some of the test techniques, such as exploratory and experience based testing, which totally rely on testers abilities and skills, are found to add quality to the test process implemented in the automotive domain. This study also implies that challenges with respect to resource constraints, such as difficulty in finding practitioners with right competence in testing who have expertise and experience in performing testing specific to automotive domain, act as a barrier to quality incorporation. The wastes identified in this context can be long delays or lack of people to perform testing activities (W3, W4). Almost 6 out of 8 studied projects lack dedicated testers.

The use of quality standards/measures (SP3) could help to arrive at a shared view of testing, and hence communication and knowledge sharing becomes easier, which is important when the number of people doing testing is scarce. An agile test approach may not automatically lead to quality incorporation, but with agile practices in place this can be possible (see [62] in SP6). The interview with the Scrum master in this study clearly indicated that when properly employed agile methods are a strength, not only provide flexibility and agility, but also quality.

The challenges related to time and cost constraints and testing techniques (C02), as well as tools and environment (C07) make it obvious that writing good tests is challenging. Automating tests could save time and improve value and benefits in testing. As documented in SP4 a variety of tools and approaches have been proposed to automate different types of tests, hence the options are manifold and which option to choose also depends on comparative analysis in the given context. To further improve on the situations teams can try to imple-

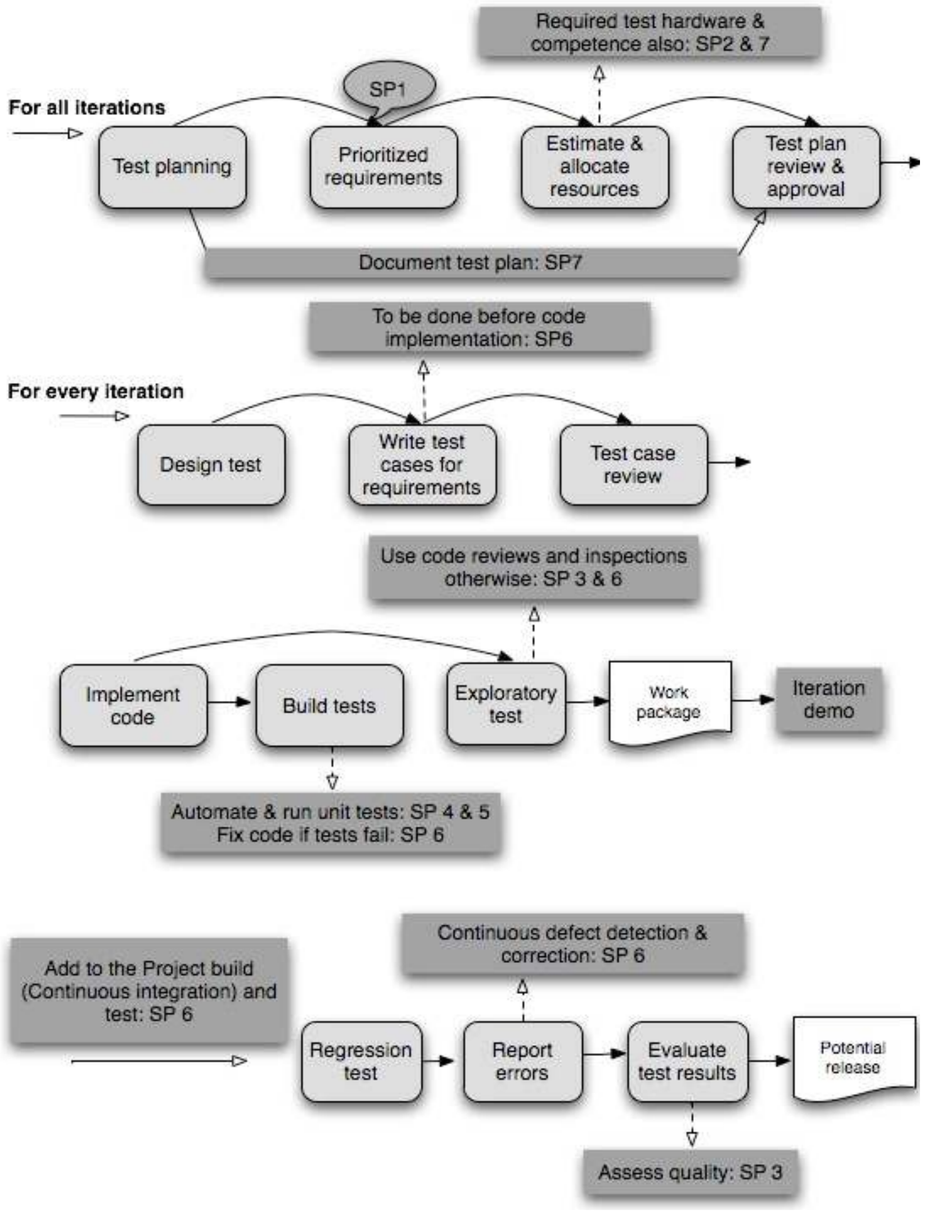
Required test hardware & competence also: SP2 & 7
For all iterations
SP1
Test planning
Prioritized requirements
Estimate & allocate resources
Test plan review & approval
Document test plan: SP7
To be done before code implementation: SP6
For every iteration
Design test
Write test cases for requirements
Test case review
Use code reviews and inspections otherwise: SP 3 & 6
Implement code
Build tests
Exploratory test
Work package
Iteration demo
Automate & run unit tests: SP 4 & 5
Fix code if tests fail: SP 6
Add to the Project build (Continuous integration) and test: SP 6
Continuous defect detection & correction: SP 6
Regression test
Report errors
Evaluate test results
Potential release
Assess quality: SP 3

Figure 5: Future State Map

ment other testing techniques, such as exploratory testing, which is already used in some projects and can find defects efficiently. Exploratory testing has been mentioned as a strength in Section 4.5.2. Automation of unit tests and regression tests can facilitate reuse of test cases and also add value to the end product. In agile development (SP6) test driven development is aiding in automation of unit tests as automated tests are written before new functionality is coded. A variety of tools to support testing, which are already used in the automotive industry, were identified and suggested based on the SLR (see SP4 in Section 5.1.3).

From this study it is reasonable to say that testing is not as emphasized as developing new code, which was also identified in [10]. Testing is given low priority, which does not facilitate knowledge sharing and knowledge transfer in testing as observed from the interviews. In this regard, competence management can be considered essential to testing with activities, which can improve skills and knowledge with respect to testing through knowledge transfer and sharing (see [32] in SP2). In addition, we believe it would be better if the required testers can be estimated in the beginning of the project and allocated in a way that they rotate and share their knowledge with a multitude of teams. This would also help them improve the competence level for every iteration and hence improve testing.

Solution proposals for the identified opportunities were based on the SLR and interviews (considering the values and benefits mentioned). The validation of the solution proposals was not possible in the the scope of this research. However, the suggested proposals were taken from peer-reviewed literature, which were validated in industry and also are well in-line with the experience of using agile in the company investigated in this study. Furthermore, the solution was presented to the practitioners who provided feedback. The future state process presented already incorporates their feedback.

7. EBSE Step 4: Evaluate and Reflect on the EBSE Process

We presented a staged EBSE process, which incorporates systematic literature review and value stream mapping, and used it for software process improvement applied to an automotive test process. In particular, our EBSE process included four steps. First, we performed a case study to investigate the challenges. Second, we performed a domain specific systematic literature. Based on that we formulated solutions proposals and linked it to our literature study findings (see Table 9). Third, we performed a value stream mapping where we mapped the challenges of the testing process to the value stream, which are all the actions needed to bring the product through the main steps of process to the customer (see Figure 4). This showed us the locations in the process where the waste (as the challenges in the value stream map are called) were located. We created the future state map that shows the locations where improvements needed to be made (see Figure 5). In the fourth step we reflect on the EBSE process.

As far as we can tell, our approach of integrating systematic literature review and value stream mapping in an EBSE process is novel. Both of the techniques are widely applied techniques in their respective domains, i.e. systematic literature reviews [25] are widely applied and software engineering research domain and value stream mapping [16] is a technique to do process improvement in the lean and automotive domain process improvement. Combining these two approaches can be seen as a good way to do industry academia collaboration and to transfer academic knowledge to industry.

However, this approach also has obvious challenges. As can be seen from this paper the problems experienced by the company where scattered to several different sub-areas of software engineering. Thus, had we performed a complete systematic literature review for all these challenges, we would have not been able to complete this work in reasonable time. Therefore, we performed a domain specific literature review to find the solutions that had been applied in the automotive and embedded domain only. Naturally, this leaves our knowledge of possible solutions limited, but it would have not been humanly possible to complete this work had we not done so.

A possible solution to this problem would be to use existing literature surveys as input to the solutions proposals and value stream mapping. However, the current literature surveys in software engineering are topic specific rather than problem specific, and thus we saw no possible way of using them. With topic specific literature review we mean that the current systematic literature surveys address questions like ”The effect of pair programming on software engineering?”[6], or ”What do we know about software productivity?”[5]. We see that industry would actually benefit more from problem specific literature surveys as they should address questions like ”Why testing window gets squeezed and what can we do about it?” or ”Why do we have poor customer communications and how can we improve it?”. Maybe in the future performing the later types of systematic literature reviews becomes more common if the main goal of the software engineering research community is to serve industrial needs.

8. Validity Threats

A validity threat is a specific way in which you might be wrong [64] Research based on empirical studies does have threats to consider. Potential threats relevant to this case study are: Construct validity, external validity and reliability or conclusion validity.

8.1. Construct validity:

Construct validity is concerned with obtaining right measures for the concept being studied. The following actions were taken to mitigate this threat [20].

- *Selection of people for interviews:* There is a risk to bias the results of the case study through a biased selection of interviewees. The selection of the representatives of the company was done having the following aspects in mind such as process knowledge, roles, distribution across various hierarchies and having a sufficient number of people involved (according to Table 2). Hence, care has been taken to assure variety (across projects and roles) among selected people, which aided in reducing the risk of bias.
- *Reactive bias:* There is a threat that the presence of research worker influences the outcome of the study. There has been a contract signed by the research worker and the organization to maintain confidentiality, and each interviewee received a guarantee for treating their responses anonymously and only presenting aggregated results.
- *Correct data Interview:* Construct validity also addresses misinterpretation of interview questions. Firstly, a mock-interview was conducted with an employee with the organization in order to ensure the correct interpretation of the questions. Furthermore, the context of the study is clearly explained (through mail/in person) before the interview. Member checking was done for each interview by sending the results to each interviewee to validate them.

8.2. External Validity:

External Validity is the ability to generalize the findings to a specific context as well as to general process models [20].

- *A specific company:* One of the potential threats to validity is that test process at only one company is studied for this case study. It has been impossible to conduct a similar study at another organization since this particular case study is aimed to improve the test processes at the respective organization only. However, this type of in-depth study gave an insight into automotive development in general and the findings have been mapped from the company's specific processes to general processes. Thus, the context of the study and the situation at case organization are clearly described in detail, which supports the generalization of the problems identified, allowing others to understand how the results map to another specific context.
- *Team size:* The domain studied is automotive and embedded software engineering. The team size is influencing the applicability of the solution and the challenges discussed here, e.g. small teams are a central practice of working agile [65]. We would like to get an indication whether our case is typical with respect to the population of automotive software companies. Given that we were not finding surveys or studies reporting team sizes in that domain, we looked into similar domains. Hence, we extended our search looking at the embedded domain in general (including avionics, robotics, etc.). According to the survey presented in [66] the team sizes vary a lot, from teams with less than 3 people to teams with more than 300 people. The most common cases are sizes of less than three people (8 out of 31 cases), team sizes of three to 10 people (11 cases) and sizes of more than 10 to 30 people (10 cases). For team sizes in general it was found that the size was 8.16, standard deviation 20.16 and min-max 1 to 468 (cf. [67]). The authors do not report median, but based on the numbers it is certainly less than the mean. This is because the few very large teams have a big impact on the mean and on the small team side we cannot have teams smaller than one person. Also, note the high standard deviation. Thus, the questions of typical the team size is similar to questions what is a typically size of a town or a software module [68]. They all are very likely to be distributed according to power law (or Pareto principle), i.e. there are few large teams/cities/modules and many small teams/cities/modules. Hence, our team sizes seem to match those of other (but not all) companies.

  The team sizes studied in this case relate to 19 out of 31 cases reported in the survey by Salo and Abrahamsson [66] (team sizes less than 10), which indicates that a similar domain works with smaller teams as well. Regarding the applicability of the solution (agile test process for automotive) we can only generalize to team sizes studied, and hence for automotive companies working with smaller teams, or companies breaking up a very large team into smaller teams.

8.3. Reliability:

This threat is concerned with repetition or replication, and in particular that the same result would be found if re-doing the study in the same setting [20].

Interpretation of data: There is always a risk that the outcome of the study is affected by the interpretation of the researcher. To mitigate this threat, the study has been designed so that the data is collected from different sources, i.e., to conduct triangulation to ensure the correctness of the findings. The interviews have been recorded and the correct interpretations have been validated through member checking to improve the reliability of data. With respect to the structure of the results (coding, identification of challenge areas) the researchers participating in the study reviewed the coding and interpretation to avoid researcher bias. We also presented the results to the studied company, who agreed to the structuring and the identified results. Company representatives in addition to that reviewed the article to check if the information is correct with respect to their experience. Prior to reviewing the report, we also created a structure of the results as a mind-map. This mind-map was also used for review/member checking. We had one of the practitioners do coding as well, to verify if we would arrive at the same interpretation, which allowed us to discuss/refine the analysis and hence increase the soundness of interpretation.

9. Discussion

9.1. RQ1: What are the practices in testing that can be considered as strengths in automotive domain?

The strengths of the testing were first listed in Section 4.5.2 and further elaborated in Section 6 where they were mapped general value producing activities (see Table 11). Working in small agile teams was considered as a benefit as it reduces the need for documentation and bureaucracy. Small teams were also perceived to lead to more iterative development, easier continuous integration and allowing a better alignment of testing with software requirements and design. Furthermore, team size and the use of agile methods were also linked by the interviewees to the improved communication that made software testing easier. A prior works also describe the benefits of small and agile teams in relation to software testing [69]. Additionally, the importance of good communication has been repeatedly discussed in the software engineering literature [70, 71].

The shared role of having the same person to write code and test for that code was considered as a benefit in a small, but the viewed was a drawback in large teams. In many cases, there were no dedicated testers either in small or large teams. Traditionally, the software testing literature suggests that one should not test their own programs [72]. However, a survey of unit-testing practices in industry actually shows that the developers create the unit test [73] and not by an outside test-organization as suggested for example in [72]. Furthermore, a case study of three software product companies shows a similar low share of dedicated testers [74] as we have reported in this paper. Our findigs extend the findings of the prior work by suggesting that the need for dedicated testers and the question whether one should test their own programs might be related to the context variable of the team size.

However, also large teams experienced several benefits that were not identified in small teams. For example large teams had often experienced people available. This allowed using testers knowledge and skill and in deciding which test to execute. A recent work of studying exploratory testing in the industry highlights the importance of tester's knowledge [75], as does another study of test design also coming from the industry [76]. Our finding strengthens the limited prior evidence of the role of knowledge in industrial software testing.

Exploratory testing was found to be a strength, which is a good complement to scripted and automated testing. There is evidence of benefits of ET from industrial context (cf. [75]), such as being able to find the most critical defects. Also an experimental comparison between ET and TCT suggests that test cases may not add any benefits when considering defect detection effectiveness [77].

Large teams also had benefits from better management that was visible in the reuse of testing artefacts, better organization of test activities, more organized tool usage, and controlling exploratory testing with session based management. So although, considerable benefits were seen stemming from small team and agile way of working also the large teams had benefits, but they originated more from traditional management.

9.2. RQ2: What the challanages / bottlenecks identified in testing automotive systems?

Even though many of the large team benefits came from better management as pointed out in the previous section, it was also found that organization and process issues were problematic in both and large and small teams. Lack of an unified testing process was found problematic. Similar challenges on general software process improvement can be found, e.g. people are not aware of the process or the process is incompatible. We also found haste in testing that was cause by a squeezed testing window due to delays in software development. The time and cost constraints were also closely linked to the process challenges, e.g. if a customer is not able to provide validation requirements then testing is obviously difficult to scope and manage. In the gray literature procured by industry consultants, it is reported that such squeezing of the testing window can be linked backed to the V-model of software development [78] . Furthermore, stakeholders poor attitudes towards testing are something that has repeatedly been mentioned in presentation and discussion as the authors have interacted with several software testing professionals.

Additionally, the human resources constrain to testing was found in teams without dedicated testing team. Thus, they would have needed dedicated testing personnel or in general, more personnel that someone would have had time for creating executing tests. The same problem was found in prior work investigating companies where testing was purposefully organized as a crosscutting activity rather than relying on specialized testers [74].

We found two types of knowledge related problems in software testing. First, problems were related to the domain or to the system under test. In other words, the new testers in the case company needed an training or experience before they could make useful contributions. The prerequisite of domain and system knowledge was particularly linked to exploratory testing that matches recent findings on exploratory testing [75]. We also found that lack of appreciation to software had led to lack of knowledge regarding testing fundamentals. The lack of testing fundamentals has also been recognized by [79] who indicates that although experienced industry professionals know basic testing techniques they may not be able to apply them correctly. Again, our empirical findings strengthen our knowledge of the problems of industrial software testing and it seems that lack of company specific knowledge as well as lack of fundamental testing knowledge are challenges also in the automotive domain.

Problems related to requirements were mentioned in three development teams. It is well understood that well specified requirements form the bases for software testing, but addressing this problem in practice has until recently received limited attention in empirical software engineering research [80, 81]. In our case, we found problems related to requirement clarity, volatility and traceability.

The communication challenges were related either due to lack of customer interaction regarding to the software requirements or due interaction of previous project employees who had been transferred to other projects before testing. It is natural that lack of customer communication combined with insufficient requirements leads to problems in software testing. However, the project staff turnover also affects testing as the original developers or other personnel will not be available to answer developers questions towards the end of the project.

We also found challenges related to testing techniques, tools, and environment. It is surprising that our company was lacking the tools of test automation, as one would think that automated test would be well understood in the embedded domain such as automotive industry. The lack of tool usage could be traced to the improper fit between the test automation tools the company had and the requirements for such tools. Sometimes the company was even forced to develop their own tools. The problems with the tools are not surprising as a recent survey indicated that roughly half of the respondents considered that the current testing tools available in the market offer a good fit for their needs [82].

Also incorporating quality aspects was considered problematic, e.g. reliability goals were seen as difficult achieve. Furthermore, missing or too late definition of quality goals and lack of measures of quality was perceived problematic. It is not surprising that companies face problems in these areas and only in recent years have light weight methods, which have been industrially validated been developed to answer such problems [13, 83].

Problems related to fixing were also found as it was indicated that finding defects in the source code is difficult from a complex system. Other reason for difficult defect detection was a big bang integration and testing at the end of the project rather than continuous testing and integration during the release.

A dualistic problem was faced with regarding the documentation of testing. On the one hand it was claimed that documentation was insufficient. On the other hand it was claimed that there is too much documentation that does not support software testing activities in the company, which was partially due to the poor updating of the documents. These documentation related issues are quite common in software industry and partly the reason why agile methods have taken over - when there is no documentation one does not have to feel disappointed when it is constantly outdated.

9.3. *RQ3: What improvements for the automotive testing process based on practical experiences were suggested in the literature?*

For 15 out of 26 challenges we found solutions that address those challenges in literature on automotive testing. Given that we scoped the literature review on literature related to automotive and embedded software engineering, we were not able to identify solutions for all challenges in the automotive literature. Hence, the solutions might be available beyond the scope of our review, but they were not applied in the studied domain. We identified seven solution proposals based on the literature, which were related to requirements management, competence management, quality assurance and standards, test automation and tools, agile incorporation, and test management. The overview of the solutions is presented in Section 5.1.3, and the mapping between challenges and solution references in Table 9.

9.4. *RQ4: What is value and waste in the process considering process activities, strengths and weaknesses identified in EBSE Step 1?*

We identified wastes and mapped their locations to the automotive software testing process used in the company. A consolidated view of the wastes and values is presented in Table 13. The table reveals that many wastes are due to requirements issues, highlighting the importance of requirements in software testing. Wastes W2, W3, W5, and W6 are related to requirements issues. A consequence of this is the recent focus of research on aligning requirements research with verification research. The importance of combining both disciplines is, for example, highlighted in [84].

9.5. *RQ5: Based on the solutions identified in EBSE Step 2, how should the process represented by the current value stream map be improved?*

A new process was proposed that incorporates the improvement proposals from the literature review (see Figure 5). The process incorporates agile software development, reviews, automation of tests, as well as continuous defect detection and correction. It was visible that the process of testing is not only concerned by the improvements suggested, but also the requirements process is affected. Overall, we can conclude that it is important to conduct an impact assessment of the improved process on other parts of the process to align the improvement efforts. That is, when the process is updated we have to think about the other process, but also how the change affects the organization, architecture, and so forth. In this regard, literature talks about alignment of the aspects of business, organization, process, and architecture (BAPO), but to date no solutions for the systematic alignment of those activities are not available [85]. Hence, we highlight the importance here, but were not able to provide a solution for the end-to-end process at this point.

The practitioners reviewed the process, and agreed on its design, feasibility and also that it has the potential to address the challenges raised in the company context.

9.6. *RQ6: What was working well in using the EBSE process with mixed research methods and how can the process be improved?*

In our research we were facing a situation to improve a process with scattered problem areas (e.g. requirements, test automation, communication issues, etc.) and at the same time having an expectation from our industry collaborator to provide a solution for their problem in a reasonable time. As a consequence we decided to scope the literature review focusing on automotive software engineering. In a longer term we found that existing literature reviews would help that are less general, and more problem driven/focused. We provided two examples, namely, *"Why testing window gets squeezed and what can we do about it?"* or *"Why do we have poor customer communications and how can we improve it?"*.

Beyond that we also see a need to further extend and learn about evidence-based approaches building upon the previous research. There are a variety of strategies available to conduct the steps of the evidence-based process, one way of conducting the evidence-based process for process improvement has been presented here. In the future, we would largely benefit from contrasting different strategies and providing evidence of their impact on the result of e.g. a literature review. This will also help in making trade-off decisions between effort/time invested in the research, and the quality of the output, allowing us to elaborate to companies how our strategies will impact what we propose for them. Example questions are:

- Is it better to search for articles using search strings, conduct snowball sampling (looking at references of identified papers - backward snowballing; or looking at papers citing a paper identified - forward snowballing)?
- Do we have to find all articles, or is there a good strategy of sampling so that the overall conclusion of a systematic review does not change?

- How can we select studies in an unbiased manner efficiently to solve our research problem?
- How shall we interpret and aggregate the conclusions of different studies?

In future studies on EBSE we will track time and effort as this is an important variable, which is seldom reported (neither by us so far), but we recognize the need for that to make informed decisions of what strategy to choose. Literature that can be built upon to answer the above mentioned questions has been presented, e.g. Zhang et al. [86] evaluate searches in systematic literature reviews, Jalali et al. [87] compared snowball sampling with database searches, Ali and Petersen [88] identified paper selection strategies from a set of identified articles, and [89] present strategies to aggregate evidence.

10. Conclusions

We used a staged evidence-based process that incorporates case study, systematic literature review, and value stream analysis to study the challenges and to create a map of solution proposals for our case company. These techniques have been widely applied, but to our knowledge this is the first time they have been used in combination for solving a problem in a concrete case study. We see that combining these approaches is a good way to do industry academia collaboration as it allows studying real industrial problems with rigorous academic methods and produces a result that is mapped to the companies current software processes. However, when conducting this we realized a major challenge in this approach as well. Often the industry problems are scattered over different areas, e.g. problems affecting a testing process may stem for example from requirements engineering, knowledge management, or test environments. Performing a literature study over such a large area would be a task with huge work load. We solved this by performing a domain specific literature review where we focused only on the studies of automotive and embedded domain. Another solution would be to utilize existing literature reviews. However, they are currently topic specific rather than problem specific, which severely restricts using them off-the-shelf. Perhaps in the future, systematic literature reviews should be made problem specific, i.e. to help the industry, rather than topic specific, i.e. helping the researchers and thesis students.

For the automotive test process, we have identified the strengths and challenges of software testing in automotive software testing. We did this with a case study of single company by studying 11 different development teams of three different departments. We found that although automotive has its own set of unique challenges, e.g. issues related to testing environment, still most of the challenges identified in this paper can be linked to problems reported from other domains as discussed in the previous section. Although, one could think that automotive domain would often follow strict and rigorous software development approaches, e.g. use formally specified requirements and highly plan-driven software development processes, we found that the opposite was true. Furthermore, it was found that one of the development teams that appeared to be one of the least problematic was benefiting from agile software development methods. However, it must be admitted that the larger teams often benefitted from better management than the small teams did.

In future work there is a need to apply the evidence based process to other process improvement problems. Furthermore, we observed the need to characterize the automotive domain with respect to state of practice (e.g. regarding team size). Hence, surveys and questionnaires characterizing the domain are needed.

11. Acknowledgements

We would like to thank all the participants in the study who provided valuable input in interviews. Furthermore, we thank the anonymous reviewers for valuable comments that helped in improving the paper. This work has been supported by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government.

References

[1] B. Kitchenham, T. Dyba, M. Jorgensen, Evidence-based software engineering, in: Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on, IEEE, 2004, pp. 273–281.

[2] K. Petersen, R. Feldt, S. Mujtaba, M. Matsson, Systematic mapping studies in software engineering, in: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE 2012), British Computer Society, 2008, pp. 71–80.

[3] B. Kitchenham, Procedures for performing systematic reviews, Tech. Rep. TR/SE-0401, Department of Computer Science, Keele University, ST5 5BG, UK (2004).

[4] B. A. Kitchenham, E. Mendes, G. H. Travassos, Cross versus within-company cost estimation studies: A systematic review, IEEE Trans. Software Eng. 33 (5) (2007) 316–329.

[5] K. Petersen, Measuring and predicting software productivity: A systematic map and review, Information and Software Technology.

[6] J. Hannay, T. Dybå, E. Arisholm, D. Sjøberg, The effectiveness of pair programming: A meta-analysis, Information and Software Technology 51 (7) (2009) 1110–1122.

[7] K. Grimm, Software technology in an automotive company - major challenges, in: Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA, 2003, pp. 498–505.

[8] D. Sundmark, K. Petersen, S. Larsson, An exploratory case study of testing in an automotive electrical system release process, in: Industrial Embedded Systems (SIES), 2011 6th IEEE International Symposium on, Vasteras, Sweden, 15-17 June, 2011, 2011, pp. 166–175.

[9] A. Pretschner, M. Broy, I. H. Krüger, T. Stauner, Software engineering for automotive systems: A roadmap, in: International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA, 2007, pp. 55–71.

[10] E. Bringmann, A. Krämer, Model-based testing of automotive systems, in: First International Conference on Software Testing, Verification, and Validation, ICST 2008, Lillehammer, Norway, April 9-11, 2008, 2008, pp. 485–493.

[11] R. Feldt, R. Torkar, E. Ahmad, B. Raza, Challenges with software verification and validation activities in the space industry, in: Third International Conference on Software Testing, Verification and Validation, ICST 2010, Paris, France, April 7-9, 2010, pp. 225–234.

[12] L. Karlsson, Å. G. Dahlstedt, B. Regnell, J. N. och Dag, A. Persson, Requirements engineering challenges in market-driven software development - an interview study with practitioners, Information & Software Technology 49 (6) (2007) 588–604.

[13] B. Regnell, R. Svensson, T. Olsson, Supporting roadmapping of quality requirements, Software, IEEE 25 (2) (2008) 42–47.

[14] S. Mujtaba, R. Feldt, K. Petersen, Waste and lead time reduction in a software product customization process with value stream maps, in: 21st Australian Software Engineering Conference (ASWEC 2010), 6-9 April 2010, Auckland, New Zealand, 2010, pp. 139–148.

[15] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–164.

[16] H. L. McManus, Product development value stream mapping (pdvsm) manual, Tech. rep., Center for Technology, Policy, and Industrial Development, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, USA (September 2005).

[17] Y. Cai, J. You, Research on value stream analysis and optimization methods, in: Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on, IEEE, 2008, pp. 1–4.

[18] R. K. Yin, Case study research : design and methods, 4th Edition, SAGE, London, 2009.

[19] C. Robson, Real world research : a resource for social scientists and practitioner-researchers, 2nd Edition, Blackwell, Oxford, 2002.

[20] K. Petersen, C. Wohlin, The effect of moving from a plan-driven to an incremental software development approach with agile practices - an industrial case study, Empirical Software Engineering 15 (6) (2010) 654–693.

[21] M. Khurum, T. Gorschek, M. Wilson, The software value map - an exhaustive collection of value aspects for the development of software intensive products, Journal of Software: Evolution and Process, in print.

[22] J. M. Morgan, J. K. Liker, The Toyota product development system : integrating people, process, and technology, Productivity Press, New York, 2006.

[23] M. Poppendieck, T. Poppendieck, Lean software development: an agile toolkit, Addison-Wesley, Boston, 2003.

[24] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information & Software Technology 50 (9-10) (2008) 833–859.

[25] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. Rep. EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University (July 2007).

[26] G. Park, D. Ku, S. Lee, W. Won, W. Jung, Test methods of the autosar application software components, in: ICCAS-SICE, 2009, IEEE, 2009, pp. 2601–2606.

[27] M. Weber, J. Weisbrod, Requirements engineering in automotive development: Experiences and challenges, IEEE Software 20 (1) (2003) 16–24.

[28] G. Mueller, J. Borzuchowski, Extreme embedded a report from the front line, in: OOPSLA 2002 Practitioners Reports, ACM, 2002, pp. 1–ff.

[29] S. Islam, H. Omasreiter, Systematic use case interviews for specification of automotive systems, in: 12th Asia-Pacific Software Engineering Conference (APSEC 2005), 15-17 December 2005, Taipei, Taiwan, 2005, pp. 17–24.

[30] S. Bühne, G. Halmans, K. Pohl, M. Weber, H. Kleinwechter, T. Wierczoch, Defining requirements at different levels of abstraction, in: 12th IEEE International Conference on Requirements Engineering (RE 2004), 2004, pp. 346–347.

[31] X. Liu, X. Yan, C. Mao, X. Che, Z. Wang, Modeling requirements of automotive software with an extended east-adl2 architecture description language, in: Industrial and Information Systems (IIS), 2010 2nd International Conference on, Vol. 2, IEEE, 2010, pp. 521–524.

[32] A. Puschnig, R. T. Kolagari, Requirements engineering in the development of innovative automotive embedded software systems, in: 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan, 2004, pp. 328–333.

[33] H. Post, C. Sinz, F. Merz, T. Gorges, T. Kropf, Linking functional requirements and software verification, in: RE 2009, 17th IEEE International Requirements Engineering Conference, Atlanta, Georgia, USA, August 31 - September 4, 2009, 2009, pp. 295–302.

[34] B. Hwong, X. Song, Tailoring the process for automotive software requirements engineering, in: Automotive Requirements Engineering Workshop, 2006. AuRE'06. International, IEEE, 2006, pp. 2–2.

[35] P. Braun, M. Broy, F. Houdek, M. Kirchmayr, M. Müller, B. Penzenstadler, K. Pohl, T. Weyer, Guiding requirements engineering for software-intensive embedded systems in the automotive industry, Computer Science-Research and Development (2010) 1–23.

[36] N. Heumesser, F. Houdek, Experiences in managing an automotive requirements engineering process, in: 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan, 2004, pp. 322–327.

[37] S. Lee, T. Park, K. Chung, K. Choi, K. Kim, K. Moon, Requirement-based testing of an automotive ecu considering the behavior of the vehicle, International Journal of Automotive Technology 12 (1) (2011) 75–82.

[38] F. Merz, C. Sinz, H. Post, T. Gorges, T. Kropf, Abstract testing: Connecting source code verification with requirements, in: Quality of Information and Communications Technology, 7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010, Porto, Portugal, 29 September - 2 October, 2010, Proceedings, 2010, pp. 89–96.

[39] M. Conrad, I. Fey, S. Sadeghipour, Systematic model-based testing of embedded automotive software, Electr. Notes Theor. Comput. Sci. 111 (2005) 13–26.

[40] M. Lochau, U. Goltz, Feature interaction aware test case gener-

ation for embedded control systems, Electr. Notes Theor. Comput. Sci. 264 (3) (2010) 37–52.

[41] O. Bühler, J. Wegener, Evolutionary functional testing, Computers & OR 35 (10) (2008) 3144–3160.

[42] C. Pfaller, A. Fleischmann, J. Hartmann, M. Rappl, S. Rittmann, D. Wild, On the integration of design and test: A model-based approach for embedded systems, in: Proceedings of the 2006 International Workshop on Automation of Software Test, AST 2006, Shanghai, China, May 23-23, 2006, 2006, pp. 15–21.

[43] P. M. Kruse, J. Wegener, S. Wappler, A highly configurable test system for evolutionary black-box testing of embedded systems, in: Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009, 2009, pp. 1545–1552.

[44] J. Wegener, Evolutionary testing techniques, in: Stochastic Algorithms: Foundations and Applications, Third International Symposium, SAGA 2005, Moscow, Russia, October 20-22, 2005, Proceedings, 2005, pp. 82–94.

[45] R. Awedikian, B. Yannou, Design of a validation test process of an automotive software, International Journal on Interactive Design and Manufacturing 4 (4) (2010) 1–10.

[46] A. Brillout, N. He, M. Mazzucchi, D. Kroening, M. Purandare, P. Rümmer, G. Weissenbacher, Mutation-based test case generation for simulink models, in: Formal Methods for Components and Objects - 8th International Symposium, FMCO 2009, Eindhoven, The Netherlands, November 4-6, 2009. Revised Selected Papers, 2009, pp. 208–227.

[47] C. Schwarzl, B. Peischl, Test sequence generation from communicating uml state charts: An industrial application of symbolic transition systems, in: QSIC, 2010, pp. 122–131.

[48] K. Lakhotia, M. Harman, H. Gross, Austin: A tool for search based software testing for the c language and its evaluation on deployed automotive systems, in: Search Based Software Engineering (SSBSE), 2010 Second International Symposium on, IEEE, 2010, pp. 101–110.

[49] V. Chimisliu, C. Schwarzl, B. Peischl, From uml statecharts to lotos: A semantics preserving model transformation, in: Proceedings of the Ninth International Conference on Quality Software, QSIC 2009, Jeju, Korea, August 24-25, 2009, 2009, pp. 173–178.

[50] P. Runeson, C. Andersson, M. Höst, Test processes in software product evolution - a qualitative survey on the state of practice, Journal of Software Maintenance 15 (1) (2003) 41–59.

[51] O. Niggemann, A. Geburzi, J. Stroop, Benefits of system simulation for automotive applications, in: Model-Based Engineering of Embedded Real-Time Systems - International Dagstuhl Workshop, Dagstuhl Castle, Germany, November 4-9, 2007. Revised Selected Papers, 2007, pp. 329–336.

[52] B. Schätz, Certification of embedded software - impact of ISO DIS 26262 in the automotive domain, in: Leveraging Applications of Formal Methods, Verification, and Validation - 4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Crete, Greece, October 18-21, 2010, Proceedings, Part I, 2010, p. 3.

[53] J. Seo, B. Choi, S. Yang, Lightweight embedded software performance analysis method by kernel hack and its industrial field study, Journal of Systems and Software 85 (1) (2012) 28–42.

[54] J.-L. Boulanger, V. Q. Dao, Requirements engineering in a model-based methodology for embedded automotive software, in: 2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies, RIVF 2008, Ho Chi Minh City, Vietnam, 13-17 July 2008, 2008, pp. 263–268.

[55] J. Cha, D. Lim, C. Lim, Process-based approach for developing automotive embeded software supporting tool, in: Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on, IEEE, 2009, pp. 353–358.

[56] T. Farkas, D. Grund, Rule checking within the model-based development of safety-critical systems and embedded automotive software, in: International Symposium on Autonomous Decentralized Systems (ISADS 2007), 21-23 March 2007, Sedona, AZ, USA, 2007, pp. 287–294.

[57] F. F. Lindlar, A. Windisch, J. Wegener, Integrating model-based testing with evolutionary functional testing, in: Third International Conference on Software Testing, Verification and Validation, ICST 2010, Paris, France, April 7-9, 2010, Workshops Proceedings, 2010, pp. 163–172.

[58] E. M. Clarke, D. Kroening, F. Lerda, A tool for checking ansi-c programs, in: Tools and Algorithms for the Construction and Analysis of Systems, 10th International Conference, TACAS 2004, Barcelona, Spain, March 29 - April 2, 2004, Proceedings, 2004, pp. 168–176.

[59] Y. Papadopoulos, C. Grante, Evolving car designs using model-based automated safety analysis and optimisation techniques, Journal of Systems and Software 76 (1) (2005) 77–89.

[60] Y. Papadopoulos, M. Maruhn, Model-based synthesis of fault trees from matlab-simulink models, in: 2001 International Conference on Dependable Systems and Networks (DSN 2001) (formerly: FTCS), 1-4 July 2001, Göteborg, Sweden, Proceedings, 2001, pp. 77–82.

[61] C. Ferdinand, R. Heckmann, H. Wolff, C. Renz, O. Parshin, R. Wilhelm, Towards model-driven development of hard real-time systems, Model-Driven Development of Reliable Automotive Services (2008) 145–160.

[62] P. Manhart, K. Schneider, Breaking the ice for agile development of embedded software: An industry experience report, in: 26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom, 2004, pp. 378–386.

[63] L. Gao, Research on implementation of software test management, in: Computer Research and Development (ICCRD), 2011 3rd International Conference on, Vol. 3, IEEE, 2011, pp. 234–237.

[64] J. Li, N. B. Moe, T. Dybå, Transition from a plan-driven process to scrum: a longitudinal case study on software quality, in: Proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM 2010, 16-17 September 2010, Bolzano/Bozen, Italy, 2010, pp. 1–10.

[65] K. Petersen, C. Wohlin, A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case, Journal of Systems and Software 82 (9) (2009) 1479–1490.

[66] O. Salo, P. Abrahamsson, Agile methods in european embedded software development organisations: a survey on the actual use and usefulness of extreme programming and scrum, IET Software 2 (1) (2008) 58–64.

[67] P. C. Pendharkar, J. A. Rodger, The relationship between software development team size and software development cost, Commun. ACM 52 (1) (2009) 141–144.

[68] P. Louridas, D. Spinellis, V. Vlachos, Power laws in software, ACM Trans. Softw. Eng. Methodol. 18 (1).

[69] V. Kettunen, J. Kasurinen, O. Taipale, K. Smolander, A study on agility and testing processes in software organizations, in: Proceedings of the 19th international symposium on Software testing and analysis, ISSTA '10, ACM, New York, NY, USA, 2010, pp. 231–240. doi:10.1145/1831708.1831737.
URL `http://doi.acm.org/10.1145/1831708.1831737`

[70] H. Saiedian, R. Dale, Requirements engineering: making the connection between the software developer and customer, Information and Software Technology 42 (6) (2000) 419–428.

[71] L. Layman, L. Williams, D. Damian, H. Bures, Essential communication practices for extreme programming in a global software development team, Information and software technology 48 (9) (2006) 781–794.

[72] I. Burnstein, Practical software testing: a process-oriented approach, Springer-Verlag New York Inc, 2003.

[73] P. Runeson, A survey of unit testing practices, Software, IEEE 23 (4) (2006) 22–29.

[74] M. V. Mäntylä, J. Itkonen, J. Iivonen, Who tested my software? testing as an organizationally cross-cutting activity, Software Quality Journal 20 (2012) 145–172. doi:10.1007/s11219-011-9157-4.

[75] J. Itkonen, M. Mäntylä, C. Lassenius, The role of the tester's knowledge in exploratory software testing, IEEE Transactions on Software Engineering (accepted).

[76] A. Beer, R. Ramler, The role of experience in software testing practice, in: Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference, IEEE, 2008, pp. 258–265.

[77] J. Itkonen, M. Mäntylä, C. Lassenius, Defect detection efficiency: Test case based vs. exploratory testing, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 2007, pp. 61–70.

[78] J. Christie, The seductive and dangerous v-model, `http://www.clarotesting.com/page11.htm/` (2008).

[79] S. Eldh, H. Hansson, S. Punnekkat, Analysis of mistakes as a method to improve test case design, in: Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on, IEEE, 2011, pp. 70–79.

[80] E. Uusitalo, M. Komssi, M. Kauppinen, A. Davis, Linking requirements and testing in practice, in: International Requirements Engineering, 2008. RE'08. 16th IEEE, IEEE, 2008, pp. 265–270.

[81] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell, P. Runeson, T. Gorschek, R. Feldt, Challenges in aligning requirements engineering and verification in a large-scale industrial context, in: Requirements Engineering: Foundation for Software Quality, 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30 - July 2, 2010. Proceedings, 2010, pp. 128–142.

[82] D. Rafi, R. D. K. Katam, K. Petersen, M. Mäntylä, Benefits and limitations of automated software testing: Systematic literature review and practitioner survey, in: 7th Workshop on automated software test, 2012. AST'08. 7th IEEE (accepted), IEEE, 2012, pp. 36–42.

[83] J. Vanhanen, M. V. Mäntylä, J. Itkonen, Lightweight elicitation and analysis of software product quality goals: A multiple industrial case study, in: Software Product Management (IWSPM), 2009 Third International Workshop on, Ieee, 2009, pp. 42–52.

[84] Z. Alizadeh, A. H. Ebrahimi, R. Feldt, Alignment of requirements specification and testing: A systematic mapping study, in: Proceedings of the ICST Workshop on Requirements and Validation, Verification and Testing (REVVERT'11), IEEE, 2011, pp. 476–485.

[85] S. Betz, C. Wohlin, Alignment of business, architecture, process, and organisation in a software development context, in: Proceedings of the International Conference on Empirical Software Engineering and Measurement (ESEM 2012), Lund, Sweden, September 19-20, 2012, pp. 239–242.

[86] H. Zhang, M. A. Babar, P. Tell, Identifying relevant studies in software engineering, Information & Software Technology 53 (6) (2011) 625–637.

[87] S. Jalali, C. Wohlin, Systematic literature studies: Database searches vs. backward snowballing, in: Proceedings of the International Conference on Empirical Software Engineering and Measurement (ESEM 2012), Lund, Sweden, September 19-20, 2012, pp. 29–38.

[88] K. Petersen, N. B. Ali, Identifying strategies for study selection in systematic reviews and maps, in: Proceedings of the International Conference on Empirical Software Engineering and Measurement (ESEM 2012), Banff, Canada, September 19-20, 2011, pp. 351–354.

[89] D. Cruzes, T. Dybå, Research synthesis in software engineering: A tertiary study, Information & Software Technology 53 (5) (2011) 440–455.

Abhinaya Kasoju is an application engineer with Systemite AB. She received her Master of Science in Software Engineering (M.Sc.) from Bleinge Institute of Technology in 2011. Her interests are databases (Oracle), value stream mapping, and empirical software engineering.

Kai Petersen is an assistant professor in software engineering at Blekinge Institute of Technology, Sweden. He received his Ph.D. and Master of Science in Software Engineering (M.Sc.) from Blekinge Institute of Technology. His research interests include empirical software engineering, software process improvement, lean and agile development, software testing and software measurement. He has over 30 publications in peer-reviewed journals and conferences. He is the industry co-chair at REFSQ 2013, the 19th International Working Conference on Requirements Engineering: Foundations for Software Quality.

Mika V. Mäntylä is a post-doc researcher at Aalto University, Finland. He received a D. Sc. degree in 2009 in software engineering from Helsinki University of Technology, Finland. In 2010 he was a visiting scholar at Simula Re- search Laboratory, Oslo, Norway. In 2011-2012 he was a post-doctoral researcher at Lund Uni- versity, Sweden. His previous studies have appeared in journals such as IEEE Transaction on Software Engineering, Empirical Software Engineering Journal and Information and Software Technology. His research interests include empirical software engineering, software testing, human cognition, defect databases and software evolution..