Testing Methods Used in the Automotive Industry: Results from a Survey

Harald Altinger
Audi Electronics Venture GmbH
Sachsstrasse 20
Gaimersheim, Germany
harald.altinger@audi.de

Franz Wotawa
TU Graz, Inst. f. Software Techn.
Inffeldgasse 16b/2
Graz, Austria
wotawa@ist.tugraz.at

Markus Schurius
Audi Electronics Venture GmbH
Sachsstrasse 20
Gaimersheim, Germany
markus.schurius@audi.de

ABSTRACT

About 90 % of all the current car innovations are based on electronics and software. The software used in a car has increased a lot reaching the size of 100 Million lines of code. Therefore, testing of automotive software has become a very critical task. In this paper we report on a questionnaire survey we carried out asking for information about testing activities within the automotive industry. We report on testing and test automation methods and tools in current use. In the survey we distinguished the different fields in the automotive industry where software development happens, i.e., research, pre-development, and series development. In addition we also discuss the opinions of the survey participants to various outlook topics.

Categories and Subject Descriptors

A.1 [**Survey**]: questionnaire results; D.2.5 [**Testing and Debugging**]: Testing tools

General Terms

Survey

Keywords

testing, survey, automotive industry, tools

1. INTRODUCTION

Cars are becoming more and more complex products requiring a lot of effort and investments. J. Power [11] said that: *"Automakers are investing billions of dollars into designing and building vehicles and adding technologies that consumers desire and demand"*. In order to fulfill the customer needs, modern cars, e.g., the 2014 Audi A8, have up to 80 or 90 Electronic Control Unit (ECU) for realizing the high amount of requested functions, ranging from simple interior lightning up to Advanced Driver Assistance Systems (ADAS) like Adaptive Cruise Control (ACC). Braess and Seiffert mentioned in their book [5] software as being among the most potential innovation enabler in a car. A wide spectrum of functionality of a current car is based on software. Some of the features can only be realized by software, e.g. infotainment. Others, e.g. CO_2 reduction, are enhanced by software to implement for example cylinder on demand or a start-stop automatic of the car's engine.

According to [6] a modern premium-class car contains close to 100 Million Lines Of Code (LOC), where some parts of the involved software is safety critical (e.g. Electronic Stability Control (ESC)), and others are expected to work (e.g. infotainment) without any trouble. Power [11] stated that the number of faults per car has an important impact on the customers opinion about the car and the brand. Hence, faults in the car have to be avoided as much as possible. Because of the increasing share of software in a car, there may be an increase of faults caused by bugs in the implementation. Consequently, the number of bugs in the car's software has an increasing effect on customers opinions and thus has to be reduced. The significance of software faults can be shown through Toyota's latest recall [12] affceting 750,000 vehicles wehre the required repair action was correcting software parameter settigns for ESC and hybrid engine components Because of the still growing importance of software in the automotive industry and potential negative impacts on customers opinion on brands, it is necessary to take appropriate steps for improving the quality of software written for cars. Besides improving development processes, verification and validation activities are among the most important tasks of software development to increase the quality of the deployed program. According to [7] finding bugs on standard computer software can cost on average about $25 during development time where 85% of all bugs are found, and up to 16,000$ after release. Therefore, it is important to fix problem as early in the development phase as possible. The actual figures for software failures within cars may vary, but the ratio seems to be comparable. An Original Equipment Manufacturer (OEM) has to compensate maintenance activities on side of the car dealers. In case of a software bug the OEM has to inform the customer and requires trained personal to update the software causing huge costs per affected car. Therefore, finding bugs before deployment is an economic request.

The question, which is also the motivation behind our survey: "How will car manufacturers implement verification and validation activities today in order to keep the amount of software faults small?" The survey is based on a question-

JAMAICA'14, July 21, 2014, San Jose, CA, USA
Copyright 2014 ACM 978-1-4503-2933-0/14/07...$15.00
http://dx.doi.org/10.1145/2631890.2631891

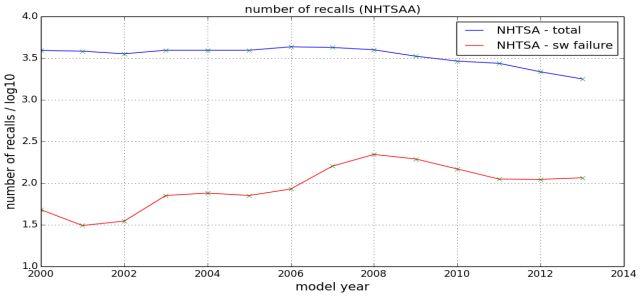

Figure 1: number vehicles affected by recalls as listed within NHTSA database

naire where we distinguish between the different fields of the automotive industry, where software development happens in practice, i.e., Research (Re), Pre Development (PD), and Series Development (SD). In the survey we asked for test methods and tools used. Moreover, we were interested in the application of test automation within the automotive industry. Before introducing the results of the survey we first discuss related research in the next section. Afterwards we present the survey results and finally, we conclude the paper.

2. RELATED RESEARCH

To our best knowledge there exists hardly any survey on the testing methods and tools used in the automotive industry. The closest we were able to find is [18] focusing on formal methods used in industrial projects. There the authors received a sample of 62 projects where 15 are related to transportation. The project size varies between 1 and 1,000 kLOC. [18] reported the use of model checking in 22 projects and automated test generation in 15 projects. It is interesting to note that correctness proofs are conducted in only 12 projects but specifications were available for 58 projects. Moreover, in the study the authors reported that formal methods had no effect on time and costs in more than 50% of the projects but improved the code quality in 92% of the cases, providing evidence for the effective use of formal methods in an industrial setting.

Zhang and Pham [19] conducted a survey with 22 responses focusing on software quality, including testing efforts, which also received some response from automotive companies. In their paper the authors state that factors like testing coverage, testing effort and testing environment have a significant impact on software reliability and that testing tools are important. Moreover, they analyzed correlations between 32 factors and ascertain testing topics like requirements and standards for carrying out the work, but not to programmers skills. One can also find white-papers, e.g. [9], [8], or success use stories, e.g. [14] in literature dealing with the use of testing methods and tools. Those publications report the use of specific tools, that originate mainly from the publishing company itself on special cases. [15] reports on Hardware in the Loop (HiL) testing and modern trends, but focus on the use of vendors products. In addition some market research has also been done, e.g. [13], but they did not list the number of participants and as common for some market research. They target the usage in past projects and made no outlook or interpretation on the data. Another important motivation behind our research relies in the information stored in certain databases listing faults and caused recalls of cars. For searching in the databases we used the words "software" or "program" in the description of fault or repair action in order to count them as a software related recall. For example, the US NHTSA database, see [10], lists 1,551 recalls, out of 49,364, caused by software failures on model years 2000 to 2014 potentially affecting 345,022,604 vehicle in USA, see figure 1. It is interesting to note that one can see a decreasing number of recalls, but an increasing rate of vehicles affected by software failures. For the model-year 2013 there is a recalled vehicle affected by a software related issue nearly every second. In summary, the NHTSA database indicate that software quality becomes more and more important for the automotive industry. From Figure 1 we can deduce that while the number of affected cars by recalls is decreasing, the number of recalls caused by software failures slightly increases. Thus we see an increasing percentage share on software related recalls, which is not a surprise when also considering the increase on the use of software in vehicles and the high potential of reusing the same software on a wide range of models.

3. THE QUESTIONNAIRE SURVEY

For our questionnaire survey we initially contacted 45 persons related to testing activities in the automotive industry with the following distribution: OEM 53% [1], Supplier (Tier 1-3), engineering service provider, software vendor 40% [2], university/research facility 7% [3]. We further asked to forward our questionnaire to specialized departments and partners that mainly focus on testing and requirements engineering. This may be a thread to the validity of our data. Persons were selected due to knowing their relation to testing and not to be a representative share of all testing activities among the automotive industry. The survey has been set up to be anonymous, so we can not eliminate duplicated answers. In order to implement the survey we made use of SurveyMonkey® to host the questionnaire and to guarantee strict anonymity for the survey participants. All original questions can be access via [3], the raw data can be requested via the authors. The survey was open from 1^{st} of February to 4^{th} of April 2014. The full survey contained 24 questions and targets the use of methods and tools in the respective departments. There were no limitations to testing methods, we included online/offline, static/dynamic, manual/automated, etc. We did allow multiple answers for all questions. Finally, we received 68 answers with the following distribution: OEM 37.74%, Supplier (Tier 1-3), engineering servie provider, software vendor 52.83%, University / Research facility 9.43%. One question was about the job description. From the obtained answers 41.1% came from software developers, 9.6% from requirement engineers, 20.5% from test engineers, 23.3% from managers, and 5.5% from others (mainly researchers). Regarding the different fields, which correspond to the three development stages Re, PD, and SD we obtained the following distribution of answers: 17.92% Re, 42.45% PD, and 39.62% SD. In the rest of this section, we discuss the obtained results in more detail.

[1] 10 companies: AUDI, BMW, Daimler, Fiat, Ford, Mitsubishi, Scania, Toyota, Volvo, VW

[2] 16 companies, among them Bosch, Continental and Magna

[3] 3 companies: KTH Royal Insitute of Technology, Graz Univesity of Technology, Frauenhofer ESK

Table 1: specification methods in use

	Re	PD	SD
	[%]	[%]	[%]
natural language	26,32	22,29	24,18
reduced vocabulary	8,77	12,05	10,46
formal language	8,77	9,64	10,46
pseudo code	7,02	8,43	7,84
functional diagrams	15,79	15,06	14,38
state diagrams	17,54	16,27	18,30
sketches	15,79	16,27	14,38

Table 2: tools in use to write specifications

name of tool	vendor	[%]
Doors	IBM	37,62
Word	Microsoft	16,83
Integrity	PTC	12,87
Excel	Microsoft	10,89
Enterprise Architect	SparxSystems	6,93

3.1 Tools in use

One main concern in this survey was to find out, which tools are in use during the requirements engineering and testing phase. First, we asked about the languages used for specifications where we distinguished *natural language*, natural language with a *reduced vocabulary*, *formal language* (i.e. logical or algebraic languages), *pseudo code*, *functional diagrams*, *state diagrams*, and *sketches*. In Table 1 we give the obtained results. Roughly a third of all specifications is textual composed (natural language and reduced vocabulary), and almost half of the specifications contains graphical elements like sketches or diagrams. The survey results for this question show no noticeable difference between Re, PD and SD.

Second, we asked about tools used during test design and the test execution phase. In Table 4 we give a list of answers with more than 2% mentioned. ExACT, EXAM, TPT, ADTF Testing framework, Tessy, dSpace HIL Test, cTest, IAV-Testmanager, LabCar Automation and CarMaker HIL Testmanager are tools for automated test case execution. They offer graphical editors to design the tests and setup the execution. These tools also generate test reports. Jenkins is an open source build server, which can be enriched using plug-ins, e.g. automated unit testing during builds. TargetLink and MATLAB/Simulink offer code generators, which can inject a test adapter and handle automated test case execution. Polyspace, QA-C, CTC++, Coverity, Understand C, VectorCast and MXAM are static code (or model) analysis tools offering code quality checkers, coding guideline analysis, bound checkers, pointer analysers, etc. AutomationDesk, SUMO, CANoe, MERAN, Messina, CTE, ECU-Test, Integrity and Rhapsody offer graphical test design and

Table 4: test automation tools

name of tool	vendor	[%]
Exact	AEV	10,14
Polyspace	Mathworks	10,14
EXAM	MicroNova	8,7
self developed		8,7
TPT	PikeTec	7,25
CANoe	Vector Informatik	5,8
QA-C	QA Systems	5,8
Integrity	PTC	2,9
CTC++	Verifysoft	2,9
Jenkins	OpenSource	2,9

Table 5: tool usage

	Re	PD	SD
	[%]	[%]	[%]
static code anlaysis	26,09	19,75	22,92
code review	21,74	16,05	12,50
code coverage	13,04	22,22	25,00
formal methods	17,39	7,41	5,21
test automation	21,74	34,57	34,38

-management. MODENA (AAMU), CANoe and CarMaker HiL Testmanager offer test environments and simulation capabilities. VectorCast and Cantata++ offer to design and automatically execute unit tests. MATELO is designed to create a test model and automatically derive test cases.

Third, we asked about the tool application area. For results have a look at Table 5. Static code analysis is used to detect array boundary violations, race conditions, memory leaks, etc. Those kind of tools are mostly executed during compile time, e.g. VectorCast offers such features. Peer review is used to discuss code with other developers, e.g. QA-C offers those methods. Code coverage is used to determine the share of code covered by test cases, e.g. with Coverity one can perform tests and determine the coverage. Formal methods are a set of mathematical techniques mainly used during specification and the modeling phase for proving correctness. Test automation is used to automatically execute tests. These tools often offer methods to design and analyze the tests, e.g. TPT. Finally, we asked about the licenses in use, see Table 6. We distinguish development frameworks, development environments, and testing tools. Many answers see scripting languages, e.g. python, as open source and not as self developed, which may be the reason for the high percentage of open source testing tools.

3.2 Differences between Re, PD, and SD

Table 5 shows the highest usage of code coverage analysis in SD. PD and SD use much more test automation than Re. This data correlates with Table 7, where Re uses a very

Table 3: specification tool kind in use

	Re	PD	SD
	[%]	[%]	[%]
Requirement management system	28,00	46,27	57,35
Word processing (any editor)	52,00	34,33	26,47
Diagram editor	20,00	19,40	16,18

Table 6: licenses in use

	development	testing
	[%]	[%]
Commercial	51,85	21,32
Open source	24,44	47,79
Self developed	23,70	30,88

Table 7: test automation methodes in use

	Re	PD	SD
	[%]	[%]	[%]
test case generation	7,69	8,60	8,16
Automated testing	15,38	32,26	35,71
Test-suite management	11,54	19,35	18,37
Manual testing	65,38	39,78	37,76

Table 8: test methodes in use

	Re	PD	SD	TE
	[%]	[%]	[%]	[%]
fuzzy testing	0,00	3,37	3,37	2,96
model based testing	32,26	32,58	35,96	38,52
mutation testing	3,23	3,37	3,37	2,96
random testing	9,68	8,99	6,74	7,41
test driven dev.	25,81	20,22	15,73	19,26
unit testing	29,03	31,46	34,83	28,89

high rate of manual testing, and Table 11, where in Re a tester is in half of the cases equivalent to the developer. Our obtained figures are comparable for Re to those reported by [13]. Accordingly to Table 7 the use of automated test case generation is rather low in all three development stages. In test case execution the situation is different. SD and PD do execute test-cases automatically whereas in Re tests are carried out manually. This observation may corresponds to faster changing requirements during Re compared to the other stages. When considering data from Table 10 and Table 13 one can see a high development focus on the whole vehicle and simulation of Re, and a high concentration on single units ("in the loop" methods and ECU hints) in SD.

Comparing testing approaches for all 68 responses with TEsting related department (TE) (see Table 8), it is easy to observe that model driven testing or model based testing" is commonly used. This may result because of the high usage of model based development (see Table 12), where MATLAB/Simulink covers a third of all development languages. Fuzz testing may be seldomly used due to very precise specifications where there is hardly any unspecific input value. Mutation testing requires code access, which is (seed Table 9) in SD for up to 60% of the cases not fully possible.

Regarding code access we give the results in Table 9. We define "Black box" as no code access, "White box" as full code access and "Grey box" as access to partial algorithm description and functional diagrams, but no access to the actual code. There we see that Re has a high access to the source code. This is due to the fact that Re mainly develops code instead of purchasing modules from other vendors. SD has the lowest code access rate. All three development stages have full access to interface descriptions.

When referring to Table 7 manual testing is (especially

Table 9: code access by development area

	Re	PD	SD
	[%]	[%]	[%]
White box	50,00	42,31	39,47
Black box	21,43	26,92	34,21
Grey box	28,57	30,77	26,32

Table 10: testing point in time

	Re	PD	SD
	[%]	[%]	[%]
Simulation	28,85	23,89	14,41
MiL	15,38	14,16	16,22
SiL	17,31	19,47	23,42
PiL	0,00	3,54	10,81
HiL	17,31	22,12	21,62
whole vehicle	21,15	16,81	13,51

Table 11: person to perform tests

	Re	PD	SD
	[%]	[%]	[%]
developer	54,29	35,53	25,71
specifier	14,29	17,11	18,57
3rd party	31,43	47,37	55,71

for Re) among the most used testing method. Automated test case execution is also often used whereas automated test case generation has still a very low usage, which is consistent with the list of tools (see Table 4), where the majority of tools support automated test case execution but only one tool is able to generate test cases. The numbers for automated test case execution correlate with those for test automation. This can be seen in Table 5. It is also interesting to have a look at testing considering the different methods of testing in the automotive industry, which correlate to the point of time of testing during the development. All "in the loop" methods use a similar approach. The System Under Test (SUT) runs within a simulation environment and is triggered via its interfaces with real live data. Model in the Loop (MiL), uses e.g. a MATLAB/Simulink Model, Software in the Loop (SiL) uses compiled software for the target machine, but runs within an emulator hosted on the development machine, Processor in the Loop (PiL) uses the SiL Software and the target processor (e.g. on an evaluation board) but not the contemplated ECU. HiL uses the target ECU with it specified hard wired interface and a simulation input values via defined signals (I/O and Bus messages). In Table 10 we depict the obtained results from our survey. We see testing based on models, software or simulation has a large share in almost all development areas. It is also interesting to note that simulation is more important for Re and PD than for SD. In Table 11 we list information regarding the persons performing the tests. As already mentioned before, in Re the developer and the tester often seems to be the same person. In contrast, SD is following the process definition where developer and tester have to be separated persons. Engineering standards, e.g. [2], require different personnel to be available.

Let us summarize the findings from the obtained data. The main differences are between data belonging to Re and SD. PD is in most cases somewhere in between. This fits very well the definition of those three development areas. It seems that in Re the main focus is on providing prototypes to show the principles where test automation seems to be less required. This is in contrast to SD and also PD where testing and test automation is a must. Moreover, in these development areas the departments focus more on one part than on the whole system.

Table 12: languages in use to write software, note formal methods are listed with 0% usage

	Re	PD	SD
	[%]	[%]	[%]
imperative, native	35,00	34,69	32,98
imperative interpreted	10,00	9,18	10,64
matlab simulink	37,50	35,71	35,11
graphic programming	2,50	1,02	2,13
graphic model design	15,00	19,39	19,15

Table 13: development targets by development area

	Re	PD	SD
	[%]	[%]	[%]
Whole car (integration)	41,67	28,79	26,67
ECU (or embedded controller)	33,33	48,48	51,67
Software library	25,00	22,73	21,67

3.2.1 Development Languages used

In order to gain more information on the current development process, we also asked the participants of our survey about the programming language paradigms used. The results of this questions are in Table 12. An imperative native language, e.g., C or C++, is an imperative language that has to be compiled directly into a machine language specific for a particular processor. Such languages are commonly used in ECU development. This explains the high percentage of about one third. Java or C# are examples for imperative interpreted languages, which are used for prototype implementations. Graphic programming languages like LabView are rather uncommon compared to graphical modeling languages like UML. We list MATLAB/Simulink as a separated category, because it has formal aspects and also enables graphical programming with interpreted mathscript support. It is commonly used together with a code generator to program an ECU or rapid prototyping devices like dSpace AutoBox. In contrast to specifications, where formal languages are in use (Table 1), no traditional formal languages, e.g. ADA, B or Z, are in use during development. Abstract Interpretation can be seen as a formal method which is supported by the Polyspace tool. There is no difference between the three development areas, maybe because of sharing commonly used tools and methods with a company. When considering the development targets (see Table 13) we see that there is a focus on the whole car in Re and a focus on ECU in SD. This is in line with the observations about the testing time point depicted in Table 10, where "in the loop" testing methods are mainly focused on one unit and not on the whole car.

3.2.2 Engineering Standards in use

Accordingly to the German law, §823 Abs. 1 BGB, §1 ProdHaftG (product liability) each manufacture is liable for person injuries due to a technical defect. If this defect could not be detected by state of the art techniques, the liability is excluded. There are many standards available in the automotive industry including ISO 26262 [2], Automotive SPICE [16], ISO 9001 [1], AUTOSAR [4], and MISRA [17].

In our survey we asked which standards to be considered during the development phase or when performing tests. In Table 14 we list only those with the most answers. In the survey we allowed double nomination, which did influence the percentage share. Safety has always been the most important car property and with ISO 26262 the existing development activities have been harmonized in an international standard. Since ISO 26262 is mandatory since the end of 2011, a lot of Re work aims to implement parts of the standard more efficiently. Development projects for non safety critical driving functions do not have to follow ISO 26262 requirements concerning software development, instead they have to comply with SPICE. Concerning the topics of software development, ISO 26262 and SPICE have got a high overlap on requirements. AUTOSAR defines interfaces and abstraction layers, which is, as expected, more a concern in SD than in Re, whereas ISO 9001 is integrated into daily processes, because of this many developers do not list it as a dedicated standard. As there is an increasing number of released ECU, which are developed according to AUTOSAR, the standard gains importance.

Table 14: engineering standards in use

	Re	PD	SD
	[%]	[%]	[%]
ISO 26262	43,48	36,36	32,95
ISO 9001	13,04	16,88	15,91
SPICE	26,09	24,68	27,27
AUTOSAR	17,39	22,08	23,86

3.3 Outlook

In the final part of our survey, we asked the participants on their opinions about testing. We treated every opinion equally regardless of the job description, department, etc. We listed five different questions and asked the participants to state their accordance from "Strongly Disagree" to "Strongly Agree" on a five point scale. The first question (Q5) was whether building a model or defining a specification consumes more development time than the actual implementation. In the second question (Q4) we asked whether the department plans to invest time and money on testing in the next year. The third question (Q3) queried their opinion on time consumption when evaluating and setting up a tool-chain for testing. The fourth question (Q2) addressed the chance of Return Of Investment (ROI) (time and money) on testing, by less required maintenance due to fewer bugs after release, shorter development time due to better specifications, etc. The fifth and last question (Q1) asked whether testing is worth the effort, e.g. higher code quality, reduced number of bugs after release, etc. Q2 actually asks about economic aspects, Q1 if the high effort (mainly different working processes on testing) on more sophisticated testing methods is worth is or if e.g. simple manual testing would achieve similar output. We depict the answers to the questions in Figure 2. From the results we can conclude that there are higher costs of requirements engineering and model building than of the actual implementation. About 94% of the participants agree on the necessity of testing and that a ROI is possible. The tool support for testing activities seems to be good, due to a wide range of participants being neutral to setup and evaluation time. Furthermore the departments do not plan to invest more on testing in the next year.

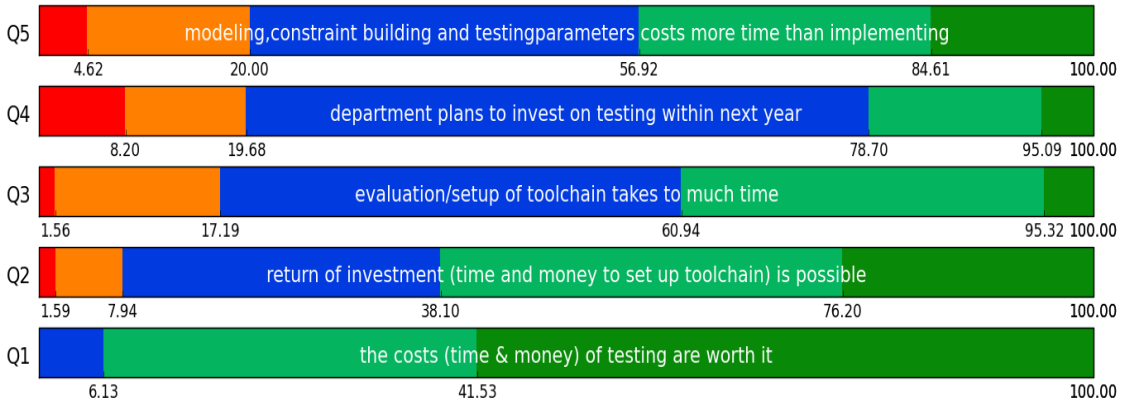

Figure 2: Opinion on testing related topic. Rating from "Strongly Disagree" (red), "Disagree" (organge), "Neutral" (blue), "Agree" (turquoise) to "Strongly Agree" (dark green). Numbers are cumulated percentage of answers.

4. CONCLUSIONS

In this paper we presented the results of a survey with the objective to gain information about the current state of testing in the automotive industry. In particular we have been interested in the degree of automation, the use of methods as well as the used tools. The presented results are based on responses of 68 participants all of them involved in software development and testing. We analyzed the results for the three development areas, Re, PD and SD, of the automotive industry. We are able to conclude that there is a difference between Re and SD. All three areas perform testing, but using different tools and methods. Re seems to address the whole vehicle, whereas SD is focused more on one unit, mainly an ECU. Moreover, in Re the developer of the software seems to be also responsible for specification and testing, whereas in SD different persons for each task are used. Automated test execution is heavily in use, but automated test case generation is not. One reason might be the fact that the currently used tools does not provide test case generation capabilities. Another observation is that formal methods are currently used in the automotive industry in the specification phase but not for testing. When considering increase of software running in a car [6], the trend of software-related recalls from Figure 1, and the explicit request of testing procedures from [2], it might be the case that the total number of failures per kLOC is already decreasing, but compensated by the increasing number of kLOC per car. Further studies are necessary to answer this and other related questions.

5. REFERENCES

[1] *ISO 9001:2008 - Quality management systems – Requirements*. International Organization for Standardization, Geneva, 2008.

[2] *ISO 26262:2011:Road vehicles – Functional safety*. International Organization for Standardization, Geneva, 2011.

[3] H. Altinger. Survey on automated testing within automotive domain, original questionnaire. `www.ist.tugraz.at/_attach/Publish/AltingerHarald/Survey_automated_testing_enu.pdf`, Mar. 2014.

[4] Autosar. Autosar. `www.autosar.org`, Mar. 2014.

[5] H. Braess and U. Seiffert. *Vieweg Handbuch Kraftfahrzeugtechnik*. ATZ/MTZ-Fachbuch. Springer Fachmedien Wiesbaden, 2013.

[6] R. N. Charette. This car runs on code. *IEEE Spectrum*, 46(3):3, 2009.

[7] W. P. Klocwork. Software on wheels. `www.klocwork.com`, Oct. 2012.

[8] H. Krisp, K. Lamberg, and R. Leinfellner. Automated real-time testing of electronic control units. `papers.sae.org`, 2007.

[9] W. P. MathWorks. Comprehensive static analysis using polyspace products. `www.mathworks.com`, Sept. 2013.

[10] U. S. D. o. T. NHTSA and USA.gov. Office of defects investigation (ODI) recalls database. `www-odi.nhtsa.dot.gov`, 1997.

[11] J. Power and M. H. Financial. 2013 J.D. power initial quality study. `autos.jdpower.com`, June 2013.

[12] T. P. Release. Toyota announces voluntary recall of certain toyota prius, RAV4, tacoma and lexus RX 350 vehicles. Dec. 2014.

[13] C. Rommel and A. Girard. Automated defect prevention for embedded software quality. `alm.parasoft.com`, Feb. 2012.

[14] M. Schneider, P. Merkle, and K. Hahmann. Test bench for complex ECU networks. `www.vector.com`, Oct. 2013.

[15] H. Schuette and M. Ploeger. Hardware-in-the-loop testing of engine control units-a technical survey. `papers.sae.org`, 2007.

[16] A. SIG. Automotive SPICE. `www.automotivespice.com`, May 2010.

[17] The Motor Industry Software Reliability Asso- and ciation. *MISRA-C:2004 - Guidelines for the use of the C language in critical systems*. MISRA, Warwickshire, 2 edition, 2004.

[18] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal methods: Practice and experience. *ACM Comput. Surv.*, 41(4), Oct. 2009.

[19] X. Zhang and H. Pham. An analysis of factors affecting software reliability. *Journal of Systems and Software*, 50(1):43–56, 2000.