Testing embedded software: A survey of the literature

Garousi, V., Felderer, M., Karapıçak, Ç. M., & Yılmaz, U. (2018). Testing embedded software: A survey of the literature. *Information and Software Technology*, *104*, 14-45. https://doi.org/10.1016/j.infsof.2018.06.016

Published in:
Information and Software Technology

Document Version:
Peer reviewed version

This is the pre-print of a paper that has been published: http://dx.doi.org/10.1016/j.infsof.2018.06.016

Testing embedded software: a survey of the literature

Vahid Garousi
Information Technology Group
Wageningen University, Netherlands
vahid.garousi@wur.nl

Michael Felderer
University of Innsbruck, Innsbruck, Austria &
Blekinge Institute of Technology, Sweden
michael.felderer@uibk.ac.at

Çağrı Murat Karapıçak
Kuasoft Information Technologies A.Ş., Ankara, Turkey
cmkarapicak@kuasoft.com
Informatics Institute, Middle East Technical
University (METU), Ankara, Turkey
murat.karapicak@metu.edu.tr

Uğur Yılmaz
ASELSAN A.Ş., Ankara, Turkey
uguryilmaz@aselsan.com.tr
Department of Computer Engineering
Hacettepe University, Ankara, Turkey
ugur.yilmaz@cs.hacettepe.edu.tr

Abstract:

Context: Embedded systems have overwhelming penetration around the world. Innovations are increasingly triggered by software embedded in automotive, transportation, medical-equipment, communication, energy, and many other types of systems. To test embedded software in an effective and efficient manner, a large number of test techniques, approaches, tools and frameworks have been proposed by both practitioners and researchers in the last several decades.

Objective: However, reviewing and getting an overview of the entire state-of-the-art and the –practice in this area is challenging for a practitioner or a (new) researcher. Also unfortunately, as a result, we often see that many companies reinvent the wheel (by designing a test approach new to them, but existing in the domain) due to not having an adequate overview of what already exists in this area.

Method: To address the above need, we conducted and report in this paper a systematic literature review (SLR) in the form of a systematic literature mapping (SLM) in this area. After compiling an initial pool of 588 papers, a systematic voting about inclusion/exclusion of the papers was conducted among the authors, and our final pool included 312 technical papers.

Results: Among the various aspects that we aim at covering, our review covers the types of testing topics studied, types of testing activity, types of test artifacts generated (e.g., test inputs or test code), and the types of industries in which studies have focused on, e.g., automotive and home appliances. Furthermore, we assess the benefits of this review by asking several active test engineers in the Turkish embedded software industry to review its findings and provide feedbacks as to how this review has benefitted them.

Conclusion: The results of this review paper have already benefitted several of our industry partners in choosing the right test techniques / approaches for their embedded software testing challenges. We believe that it will also be useful for the large world-wide community of software engineers and testers in the embedded software industry, by serving as an "index" to the vast body of knowledge in this important area. Our results will also benefit researchers in observing the latest trends in this area and for identifying the topics which need further investigations.

Keywords:

Software testing; embedded systems; embedded software; systematic mapping; systematic literature mapping; systematic literature review

TABLE OF CONTENTS

1 INTRODUCTION

Embedded software is computer software, written to control machines or devices that are not typically thought of as computers, e.g., cars and TV. According to recent surveys, approximately 90% of all processors are part of embedded systems, computing systems that continually and autonomously control and react to the environment [1]. The embedded system itself is an information processing system that consists of hardware and software components. Nowadays, the number of embedded computing systems— in areas such as telecommunications, automotive, electronics, office automation, and military applications—is steadily growing [1].

Since software is a major component of embedded systems, it is very important to properly and adequately test the embedded software, especially for safety-critical domains such as automotive and aviation. Due to the complex system context of embedded-software applications, defects in these systems can cause life-threatening situations (e.g., in airplanes), and delays can lead to huge business losses (e.g., in consumer electronics) [2].

To test embedded software in a cost-effective manner, various test techniques, approaches, tools and frameworks have been proposed by both practitioners and researchers in the last several decades. However, reviewing and getting an overview of the entire state-of-the-art and –practice in this area is almost impossible for a practitioner or a new researcher since the number of studies is simply too many, and a reader is faced with a vast body of knowledge in this area which s/he cannot simply review and digest in a reasonable time. Also, in our interaction with multiple research partners in several countries (e.g., Canada, Turkey and Austria) [3-8], we have seen in several cases that, unfortunately, many companies often spend a lot of effort to 'reinvent the wheel' (by designing a test approach new to them, but existing in the domain) due to not having an adequate overview of what already exists in this area. Knowing that they can adapt/customize an existing test technique to their own context can potentially save companies and test engineers a lot of time and money. The other main reason why we decided to conduct the review reported in this paper was that in our recent and ongoing collaborations with our industry partners in testing embedded software (e.g., [4, 9]), our colleagues and we have constantly faced numerous challenges in testing embedded software and we were uncertain of whether certain techniques already exist or we shall develop new techniques ourselves to solve those challenges.

Although there have been state-of-the-practice papers such as [2] on embedded software engineering and 'review' papers such as [10], no paper has yet studied the entire state-of-the-art and –practice in a holistic manner, which is essential for the field of testing embedded software that is equally driven by academia and industry.

To address the above need and to identify the state-of-the-art and–practice in this area and to find out what we, as a community, know about testing embedded software, we conducted a systematic literature mapping (SLM) on the technical papers written by practitioners and researchers and we present a summary of its results in this article. Our review pool included 312 technical papers published in conferences and journals. The earliest paper [11] included in the pool was published in 1984. Previous 'review' (survey) papers such as this article have appeared in different venues on other topics, e.g., about Agile development [12], and developer motivation [13], and have shown to be useful in providing concise overviews on a given area.

By summarizing what we know in this area, our article aims to benefit the readers (both practitioners and researchers) by serving as an "index" to the vast body of knowledge in this important and fast-growing area. Our review covered the types of testing topics studied, types of testing activities, types of test artifacts generated, and the types of industries in which studies have focused on.

The remainder of this article is structured as follows. A review of the related work is presented in Section 2. We describe the study goal and research methodology in Section 3. Section 4 presents the searching phase and selection of sources. Section 5 discusses the development of the systematic map and data-extraction plan. Section 6 presents the results of the study. Section 7 summarizes the findings and discusses the benefits and limitations of the study. Finally, in Section 8, we draw conclusions, and suggest areas for further research.

2 Background and related work

2.1 Challenges in testing embedded software

To better motivate the need for this review study, we summarize the characteristics of embedded systems and explain how these characteristic raise challenges in testing embedded systems and their embedded software.

Embedded software is a computer software, written to control machines or devices that "*are not typically thought of as computers*" [1]. Embedded software is typically specialized for the particular hardware that it runs on and has time and memory constraints. The "embedded software" term is often used interchangeably with firmware, although firmware can also be applied to ROM-based code on a computer, on top of which the OS runs, whereas embedded software is typically the application software on the device in question.

A precise and stable characteristic feature is that no or not all functions of embedded software are initiated/controlled via a human interface, but through machine-interfaces instead. Manufacturers build in embedded software in the electronics of e.g. cars, telephones, modems, robots, appliances, toys, security systems, pacemakers, televisions and set-top boxes, and digital watches, for example. This software can be very simple, such as lighting controls running on an 8-bit microcontroller with a few kilobytes of memory with the suitable level of processing complexity determined, or can become very sophisticated in applications such as airplanes, missiles, and process control systems [1]. When developing and testing embedded software, special attention should be paid to issues such as limited memory, CPU usage, energy consumption, and real-time needs (if any).

Embedded software is different than conventional software systems (e.g., desktop, web, or mobile applications). The major differences are due to close integration of software and hardware in embedded systems, e.g., cars, industrial controllers, robotics and aviation. Unlike conventional software, most of interfaces in embedded systems are "non-human interfaces" [14], e.g., there is usually no (or very limited) GUI and thus observing internal state of such software is not always trivial. This raises the need for sophisticated instrumentation and probing when testing these systems [14].

Also, presence of non-human interfaces leads to further challenges in manual user-interface testing. To test embedded software, one often has to develop and utilize more special software applications, e.g., test drivers, test agents, which need to be developed to provide stimulus and capture response through the non-human interfaces of embedded systems [14]. It is also often required to emulate particular electrical signal patterns on various data lines to test the behavior of the embedded software for such inputs. This should be done using special test tools.

Furthermore, high level of hardware dependency and the fact that the embedded software is often developed in parallel with the hardware lead to several other consequences and challenges. First, there may be only few samples of the newly developed hardware, thus impacting the extent of efforts by testing teams. Second, the range of the hardware unit types to test embedded software on can be quite wide. Thus, typically the testing team has to share a very limited set of hardware units among its members and/or organize remote access to the hardware. In the second case, that means that the testing team has no physical access to the hardware at all. Such challenges have led to wide development of adoption of various simulation-based testing approach in the embedded software industry, e.g., Model-in-the-Loop (MIL) testing, Software-in-the-loop (SIL), Processor-In-The-Loop (PIL), and Hardware-in-the-loop (HIL) testing.

Another challenge in testing embedded systems is that the software may work with one revision of the hardware, and does not work with another. Another aspect is when software is developed for a new hardware, high ratio of hardware defects can be identified during the testing process. In such a case, identified defects may be related to the hardware, not only software [15].

Also, defects are harder to reproduce in embedded systems. That required the embedded testing process to gather as much information as possible for looking for the root of the defect, once it is defected. Combined with the very limited debug capabilities of embedded products, that gives testers another challenge [14].

2.2 Review of secondary studies in software testing

Since our work is a study (review) of (primary) studies, it is a 'secondary' study in the area of software testing. As the related work in large, we briefly review the secondary studies in software testing. Garousi and Mäntylä conducted and reported a SLR of secondary studies in software testing recently [16]. Via a systematic literature search, that study identified a large number of secondary studies in software testing (101 papers), which are listed in an online spreadsheet [17]. Secondary studies are usually of three types: SLM studies (also often called just systematic mapping), SLR studies and regular surveys. As a snapshot, we show a randomly-selected list of 15 of the 101 secondary studies in software testing in Table 1, five in each of the above three categories.

Table 1- A selected list of 15 of the 102 of the secondary studies in software testing (the full list can be found in [17])

Type of secondary study	Secondary study area	Year of publication	Reference
SLMs	Search-based testing for non-functional system properties	2008	[18]
	Product lines testing	2011	[19]
	Graphical user interface (GUI) testing	2013	[20]
	Test-case prioritization	2013	[21]
	Software test-code engineering	2014	[22]
SLRs	Model-based testing	2007	[23]
	Automated acceptance testing	2008	[24]
	Mutation testing for Aspect-J programs	2013	[25]
	Web application testing	2014	[26]
	Testing scientific software	2014	[27]
Regular surveys	Testing finite state machines	1996	[28]
	Regression testing minimization, selection and prioritization: a survey	2012	[29]
	Testing in SOA	2013	[30]
	Test-case generation from UML behavioral models	2013	[31]
	Test oracles	2014	[32]

By seeing a large list of 102 secondary studies in software testing, one may wonder about the "value" (benefit) of such secondary studies. Analyzing and discussing usage and usefulness of SLRs in software engineering, in general, is out of scope of our paper, but we briefly touch this topic. Kitchenham et al. [33] have discussed the educational value of SLM in the software engineering literature for the students. Usefulness of SLRs for practitioners have been studied in a number of non-SE fields, such as in disability research [34], in education research [35], and in health and social care [36].

2.3 Related works: other review studies in the area of testing embedded software

No secondary study has yet been reported in the large scope of embedded software testing. A few secondary studies in more focused areas, e.g., adherence to the DO-178B standard for critical embedded systems [37], have been reported. We were able to identify 8 such studies, as listed in Table 2. For each review study, we have included the publication year, its type and some explanatory notes. For example, [38] is an informal survey of test methods for embedded systems. [39] is a regular paper in which a comparison table of model-based testing tools for embedded systems is presented.

Table 2- Related works: other survey (review) studies in the area of testing embedded software

Paper title	Publication year	Type of review and notes	Reference
In, but not of, the system: overview of embedded systems test methods	1995	A conventional survey of test methods for embedded systems	[38]
Software testing in critical embedded systems- a systematic review of adherence to the DO-178B Standard	2011	SLR in the focused areas of adherence to the DO-178B standard for critical embedded systems	[37]
Model-based testing of embedded systems in hardware in the loop environment	2012	Table 1 of this regular papers provides a comparison table of model-based testing tools for embedded systems	[39]
Evaluation of model-based testing for embedded systems based on the example of the safety-critical vehicle functions	2012	A chapter of this thesis describes a SLR performed on Model-Based Testing (MBT) approaches that are available in the automotive domain.	[40]
A survey of model-based software product lines testing	2012	An informal survey model-based testing for embedded software product lines	[41]
A systematic literature review of test case generator for embedded real time system	2014	SLR	[42]
A review on structural software-based self-testing of embedded processors	2014	A conventional survey	[1]

Environment-model based testing of control systems: case studies	2014	A chapter of this thesis describes a SLR performed on Model-Based Testing (MBT) approaches that are available in the automotive domain.	[43]
A review on verification and validation for embedded software	2016	A conventional survey	[2]
On testing embedded software	2016	As a book chapter, this work explores the advances in software testing methodologies in the context of embedded software.	[10]

3 GOAL AND RESEARCH METHOD

In the following, an overview of our research method and then the goal and review questions of our study are presented.

3.1 OVERVIEW

Based on our past experience in SLM and SLR studies, e.g., [44-48], and also using the well-known guidelines for conducting SLR and SLM studies in SE (e.g., [49-52]), we developed our SLM process, as shown in Figure 1.

Note that we had the option of conducting a SLM, or a multivocal literature review (MLR) [53-55]. A MLR is a form of a SLM or a SLR which includes the grey literature (e.g., blog posts and white papers) in addition to the published (formal) literature (e.g., journal and conference papers). In addition to a vast formal literature in the area of testing embedded software, there is also a vast grey literature in this area. For two reasons (as discussed next), we decided to conduct a SLM study in this work: (1) We observed that many practitioners in this area are publishing their proposed approaches and experience reports as papers in the formal literature (see Section 6.4.1 and 6.4.2 for active companies in this area) and thus a SLM study can still provide insights into the state of the practice in this area to a great extent; and (2) To keep our effort level manageable. A follow-up MLR can be conducted as a future work.

We discuss the SLM planning and design phase (its goal and RQs) in the next section. Section 4 to 6 then present each of the follow-up phases of the process.

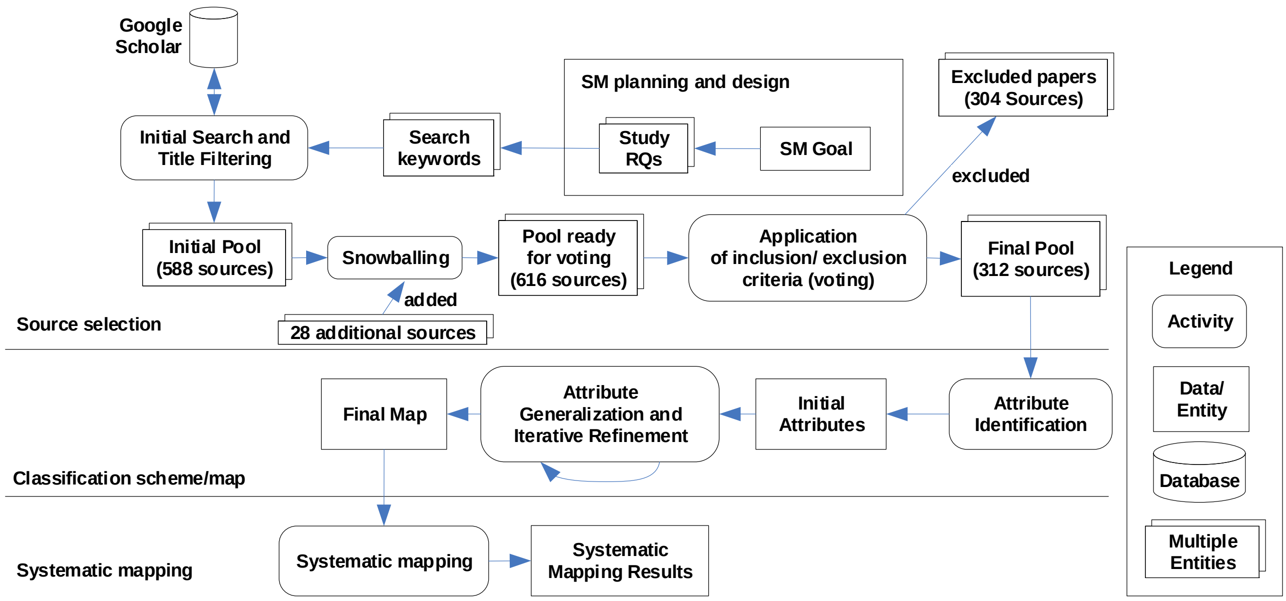


Figure 1-An overview of our SLM process (as a UML activity diagram)

3.2 GOAL AND REVIEW QUESTIONS

The goal of this study is to systematically map (classify), review and synthesize the state-of-the-art and –practice in the area of testing embedded software systems, to find out the recent trends and directions in this field, and to identify opportunities for future research, from the point of view of researchers and practitioners.

To ensure a clear focus, we defined a clear scope and boundary for our systematic literature mapping (SLM) study. We decided to only include papers on testing embedded software and exclude all the remotely-related papers, e.g., embedded software "dependability". However, we confirm that those other related areas, e.g., embedded software dependability, are important topics and there is a need for future survey studies on those topics. Based on the above goal, we raise the following review questions (RQs) grouped under three categories:

Group 1-Common to all SLM studies:

The two RQs under this groups are common to SLM studies and have been studied in previous work, e.g., [44-48].

- **RQ 1.1: Mapping of studies by contribution facet:** What are the different contributions by different sources? How many sources present test techniques/methods/methodologies, tools, metrics, models or processes? Mapping of the studies and knowing the types of contributions in them would enable us and the readers to get a high-level view of the literature's landscape based on test techniques, methods, methodologies, test tools, metrics and models.
- **RQ 1.2: Mapping of studies by research facet:** What type of research methods have been used in the studies in this area? Some of the studies presented solution proposals or weak empirical studies where others presented strong empirical studies. The rationale behind this RQ is that it is important to know and differentiate the types of research methods and the rigor used in different studies.

Group 2-Specific to the domain (testing embedded software):

- **RQ 2.1-Levels of testing:** What level(s) of testing is/are used in each study? They could be unit, integration or system testing.
- **RQ 2.2-Types of testing activities:** What types of testing activities have been conducted and proposed? Inspired by books such as [56] and our recent SLM and SLR studies such as [45, 46], we categorized testing activities as follows: test planning and management, test-case design (criteria-based), test-case design (human knowledge-based), test automation, test execution, test evaluation (oracle), and other.
- **RQ 2.3-Types of test artifacts generated:** What types of testing artifacts are generated by the test techniques proposed? After reviewing a large subset of papers and in an iterative refinement manner, we categorized them as follows: test case requirements (not input values), test case input (values), expected outputs (oracle), test code (e.g., in xUnit) and other. Let us note that test requirements are usually not actual test input values, but the conditions that can be used to generate test inputs.
- **RQ 2.4-Types of non-functional testing, if any:** In addition to functional testing, what types of non-functional testing are discussed in the paper? Note that or focus was only to include functional testing papers, but some of those papers also discussed non-functional testing aspects as well, e.g., security and load testing.
- **RQ 2.5-Techniques to derive test artifacts:** What techniques have been used to derive test artifacts? We were expecting to see techniques such as: requirement-based testing (which includes model-based testing), code-coverage analysis, risk/fault-based testing, and search-based testing.
- **RQ 2.6- Types of models used in model-based testing:** What types of models have been used in model-based testing techniques? Since we noticed that a large ratio of the studies are focused on model-based testing, we raised this RQ.
- **RQ 2.7-Testing tools (used or proposed):** What testing tools have been used or proposed in the papers? Answering this question would provide practical and useful results for practitioners.
- **RQ 2.8-Types of evaluation method:** What types of evaluation methods are used in the paper? Some papers evaluate the proposed approaches by simple examples (showing the applicability), while others use more sophisticated evaluations, e.g., coverage analysis or detecting real or artificial faults.
- **RQ 2.9-Operating systems (OS):** What operating systems (specific to embedded systems) have the papers focused on?

Group 3-Specific to system under testing (SUT): This group of RQs are specific to the SUT's studied in the papers.

- **RQ 3.1-Simulated or real systems:** Was the SUT a simulated embedded system or a real system? Since development and testing of embedded systems in real environments is not always easy or practical (e.g., the control software of a fighter jet), development and testing of those systems in simulated environments first (before real systems) are common. A popular approach in this context is X-in-the-loop development, simulation and testing: which consist of Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), Hardware-in-the-Loop (HiL), and System-in-the-Loop (SYSiL). We will discuss more on this in Section 6.

- **RQ 3.2-Number of SUTs (examples):** How many SUTs (example systems) are discussed in each paper? One would expect that each paper applies the proposed testing technique to at least one SUT. Some papers take a more comprehensive approach and apply the proposed testing technique to more SUTs.
- **RQ 3.3-Types of SUT (or example):** What are the type(s) of SUT (or example) in each paper? The SUTs in some papers are academic experimental or simple examples, while those in other papers are real open-source or commercial systems.
- **RQ 3.4-SUT programming languages:** What programming languages have the SUTs been developed in? C is usually the most popular language used for developing embedded systems. We wanted to assess this hypothesis.
- **RQ 3.5-SUT and board names:** What are the SUT and board names? It would be interesting to know the SUT and board names.
- **RQ 3.6-Application domains/industries:** We also wondered about the types of industries in which studies have focused on. While some test techniques are generic in that they can, in principle, be applied to all types of embedded software, some techniques are domain-specific. During our review, we observed these types of industries (domains): generic; home appliances and entertainment; aviation, avionics and space; automotive; defense; industrial automation /control; medical; mobile and telecom; transportation; and other.

Group 4-Demographic and bibliometric information:

- **RQ 4.1-Affiliation types of the study authors:** What are the affiliation types of the authors? We wanted to know the extent to which academics and practitioners are active in this area.
- **RQ 4.2-Active companies:** What are the active companies? It would be interesting and useful to know the active companies in this area and readers may benefit from these data, e.g., to be able to follow their upcoming works.
- **RQ 4.3-Citation analysis and highly-cited papers:** What is the citation landscape of the studies in this area, and what are the highly-cited papers in the pool? The rationale behind this RQ is to characterize how the papers in this pool are cited by other papers, to get a sense of their impact and popularity, and also to identify the papers with the highest impact in the area so that readers can benefit from them.

4 SEARCHING FOR AND SELECTION OF SOURCES

Let us recall from our SLM process (Figure 1) that the first phase of our study is article selection. For this phase, we followed the following steps in order:

- Source selection and search keywords (Section 4.1)
- Application of inclusion and exclusion criteria (Section 4.2)
- Finalizing the pool of articles and the online repository (Section 4.3)

4.1 SELECTING THE SOURCE ENGINES AND SEARCH KEYWORDS

In our review and mapping, we followed the standard process for performing systematic literature review (SLR) and systematic literature mapping (SLM) studies in software engineering. We performed the searches in both the Google Scholar database and Scopus (www.scopus.com), both widely used in review studies and bibliometrics papers, e.g., [57, 58]. The reason that we used Scopus in addition to Google Scholar was that several sources have mentioned that: "*it [Google Scholar] should not be used alone for systematic review searches*" [59] as it may miss to find a subset of papers.

All the authors did independent searches with the search strings, and during this search the authors already applied inclusion/exclusion criterion for including only those which explicitly addressed the study's topic. Our search string was: *(test OR testing OR validation OR verification) AND (embedded system OR embedded software)*.

In terms of the scope of this study, we should note that the topic of "cyber-physical systems" (CPS) is a closely related topic to embedded systems, however, after reading several online discussions among practitioners in grey literature sources such as [60, 61], we found out that: "*a CPS [may] incorporate embedded systems into itself, but the reality is that almost all embedded systems today exist outside of a CPS*" [61]. A practitioner also noted that [61]: "*CPS is a relatively new concept, embedded systems have been with us for almost half a century. So, if you have an embedded system it may or may not be part of a CPS (and currently, most are not) - but if you have a CPS, then by definition you have embedded systems as part of that CPS*". Thus, testing a CPS usually poses different challenges with respect to testing an embedded system. For the above reasons, to ensure that we would focus on clear single scope, we decided to only include "embedded systems" in our keywords, and not "cyber-physical systems". Follow-up SLM or SLR studies on testing CPSs can be conducted in future works.

In terms of timeline, our study search and selection phase was conducted in Fall 2017, and thus we included the papers published until that time.

To ensure making our paper search and selection efforts efficiency, while doing the searches using the keywords, we also conducted title filtering to ensure that we would add to our candidate paper pool only directly- or potentially-relevant papers. After all, it would be meaningless to add an irrelevant paper to the candidate pool and then remove it. Our first inclusion/exclusion criterion (discussed in Section 4.2) was used for this purpose (i.e., Does the source focus on testing embedded software?). For example, Figure 2 shows a screenshot of our search activity using Google Scholar in which directly- or potentially-relevant papers are highlighted by red boxes. To ensure efficiency of our efforts, we only added to the candidate pool those studies.

Another issue was the stopping condition when searching using the Google Scholar. As Figure 2 shows, Google Scholar provided a very large number of hits using the above keyword as of this writing (more than 2 million records). Going through all of them was simply impossible for us. To cope with this challenge, we utilized the relevance ranking of the search engine (Google's PageRank algorithm) to restrict the search space. The good news was that, as per our observations, relevant results usually appeared in the first few pages and as we go through the pages, relevancy of results decreased. Thus, we checked the first n pages (i.e., somewhat a search "saturation" effect) and only continued further if needed, e.g., when at least one result in the n^{th} page still was relevant (if at least one paper focused on testing embedded software). Similar heuristics have been reported in several other review studies, guideline and experience papers [55, 62-64]. At the end of our initial search and title filtering, our candidate pool had 531 papers (as shown in our SLM process in Figure 1).

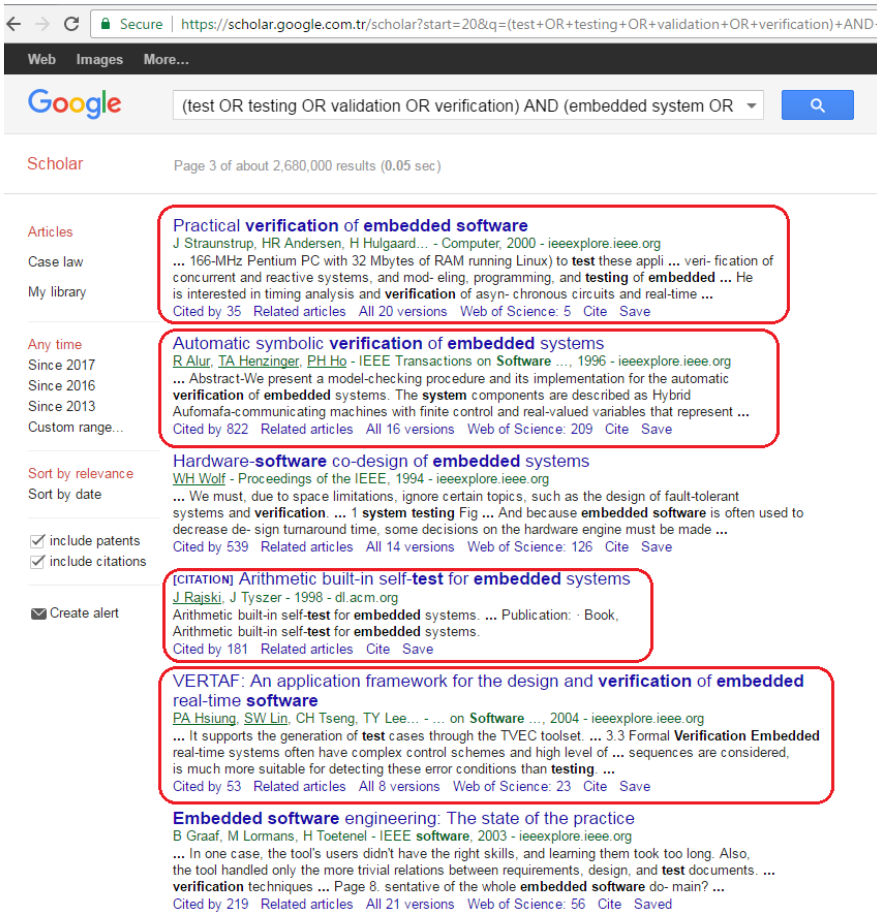


Figure 2- A screenshot from the search activity using Google Scholar (directly- or potentially-relevant papers are highlighted by red boxes)

To ensure including all the relevant sources as much as possible, we conducted forward and backward *snowballing* [50], as recommended by systematic review guidelines, on the set of papers already in the pool. Snowballing, in this context, refers to using the reference list of a paper (backward snowballing) or the citations to the paper to identify additional papers (forward) [50]. Snowballing provided 29 more papers. Some examples of the papers found during snowballing are the followings. [Source 5] was found by backward snowballing of [Source 4]. [Source 24] was found by forward snowballing of [Source 194]. Note that the '[Source i]' identifiers in this paper refer to the sources that we have included in our study's pool and can be found in an online Google spreadsheet (goo.gl/MhtbLD).

After compiling an initial pool of 560 papers, a systematic voting (as discussed next) was conducted among the authors, in which a set of defined inclusion/exclusion criteria were applied to derive the final pool of the primary studies.

4.2 Application of inclusion/exclusion criteria and voting

We carefully defined the inclusion and exclusion criteria to ensure including all the relevant sources and not including the out-of-scope sources. The inclusion criteria were as follows:

- Does the source focus on testing embedded software systems?
- Does the paper include a relatively sound validation?
- Is the source in English and can its full-text be accessed?

The answer for each question could be {0, 1}. Only the sources which received 1's for both criteria were included. The rest were excluded.

This is the pre-print of a paper that has been published: http://dx.doi.org/10.1016/j.infsof.2018.06.016

4.3 Final pool of the primary studies

As mentioned above, the references for the final pool of 312 papers can be found in an online spreadsheet (goo.gl/MhtbLD). Again, let us note that we use the format of "[Source i]" in the rest of this paper to refer to the papers in the pool as listed in the online repository (see the screenshot in Figure 3).

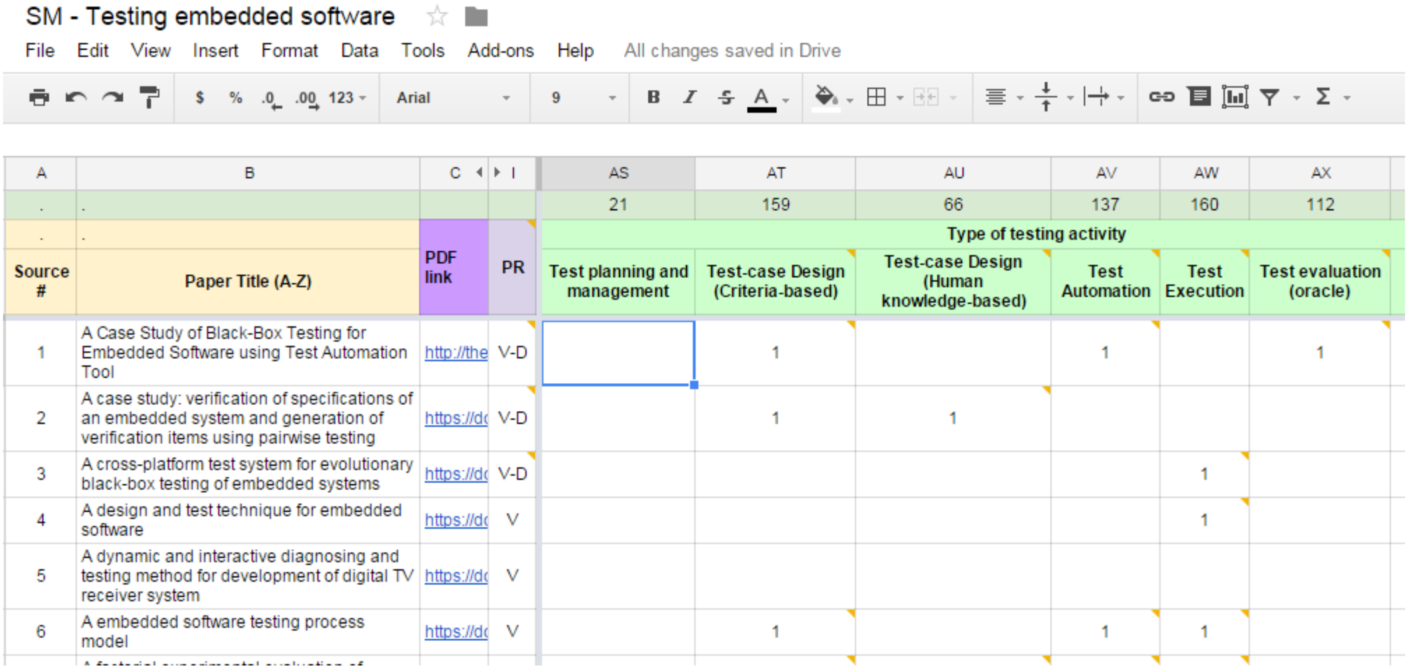


Figure 3- A screenshot from the online repository of papers (goo.gl/MhtbLD).

To visually see the growth of the field (testing embedded software), we depict the annual number of papers (by their publication years) and compare the trend with data from four other SLM/SLR studies, e.g., a SLM on web application testing [65], a SLM on Graphical User Interface (GUI) testing [20], a survey on search-based testing (SBST) [66], and a survey on mutation testing [67]. Note that the data for the other four areas are not until year 2017, since the execution and publication timelines of those studies are in earlier years, e.g., the survey on mutation testing [67] was published in 2011 and thus only has the data until 2009.

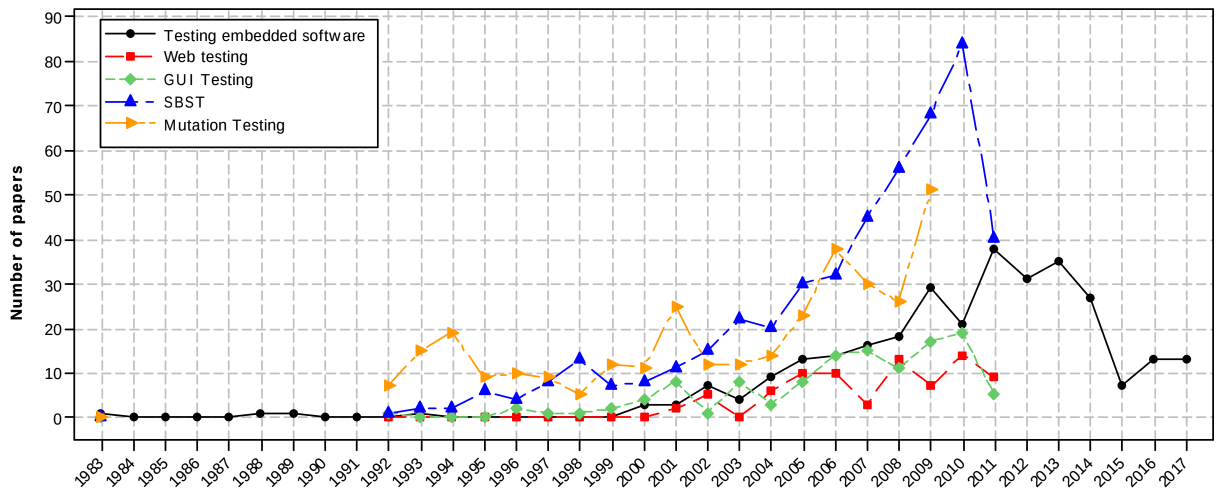


Figure 4-Growth of the field (testing embedded software) and comparison with data from four other SLM/SLR studies

5 DEVELOPMENT OF THE SYSTEMATIC MAP AND DATA-EXTRACTION PLAN

To answer each of the SLM's RQs, we developed a systematic map and then extracted data from papers to classify them using it. Details are discussed next.

5.1 DEVELOPMENT OF THE CLASSIFICATION SCHEME (SYSTEMATIC MAP)

To develop our systematic map, we analyzed the studies in the pool and identified the initial list of attributes. We then used attribute generalization and iterative refinement to derive the final map.

As studies were identified as relevant to our study, we recorded them in a shared spreadsheet to facilitate further analysis. Our next goal was to categorize the studies in order to begin building a complete picture of the research area and to answer the study RQs. We refined these broad interests into a systematic map using an iterative approach.

Table 3 shows the final classification scheme that we developed after applying the process described above. In the table, column 2 is the list of RQs, column 3 is the corresponding attribute/aspect. Column 4 is the set of all possible values for the attribute. Column 5 indicates for an attribute whether multiple selections can be applied. For example, in RQ 1.1 (research type), the corresponding value in the last column is 'S' (Single). It indicates that one source can be classified under only one research type. In contrast, for RQ 1.2 (contribution type), the corresponding value in the last column is 'M' (Multiple). It indicates that one study can contribute more than one type of options (e.g. method, tool, etc.).

Contribution type and research type classifications in Table 3 were done similar to our past SLM and SLR studies, e.g., [44-48], and also using the well-known guidelines for conducting SLR and SLM studies, e.g., [49-52]. Among the research types, the least rigorous type is 'Solution proposal' in which a given study only presents a simple example only (or proof of concept). Empirical evaluations are grouped under two categories: weak empirical studies (validation research) and strong empirical studies (evaluation research). The former is when the study does not pose hypothesis or research questions and does not conduct statistical tests (e.g., using t-test). We considered an empirical evaluation 'strong' when it has considered these aspects. Explanations (definitions) of experience studies, philosophical studies, and opinion studies are provided in Peterson et al.'s guideline paper [51].

By reviewing several software testing books [56, 68, 69] on software testing and as we had done in our previous SLM studies, e.g., [65], we classified the types of testing activities into eight types:

1. Test-case design (criteria-based): Designing test suites (set of test cases) or test requirements to satisfy coverage criteria, e.g., line coverage.
2. Test-case design (based on human expertise): Designing test suites (set of test cases) based on human expertise (e.g., exploratory testing) or other engineering goals.
3. Test scripting: Documenting test cases in manual test scripts or automated test code
4. Test execution: Running test cases on the software under test (SUT) and recording the results
5. Test evaluation: Evaluating results of testing (pass or fail), also known as test verdict
6. Test-result reporting: Reporting test verdicts and defects to developers, e.g., via defect (bug) tracking systems
7. Test automation: Using automated software tools in any of the above test activities
8. Test management: Encompasses activities related to test management, e.g., planning, control, monitoring, etc.
9. Other test engineering activities: Includes activities other than those discussed above, e.g., regression testing, and test prioritization.

Table 3: Systematic map developed and used in our study

Group	RQ	Attribute/Aspect	Categories/metrics	(M)ultiple/ (S)ingle
Group 1- Common to all SLM studies	1.1	Contribution type	{Method (technique), tool, metric, model, process, empirical results only, other}	M
	1.2	Research type	{Solution proposal (simple examples only), weak empirical study (validation research), strong empirical study (evaluation research), experience studies, philosophical studies, opinion studies, other}	S
	2.1	Levels of testing	{Unit testing, integration testing, system testing}	M

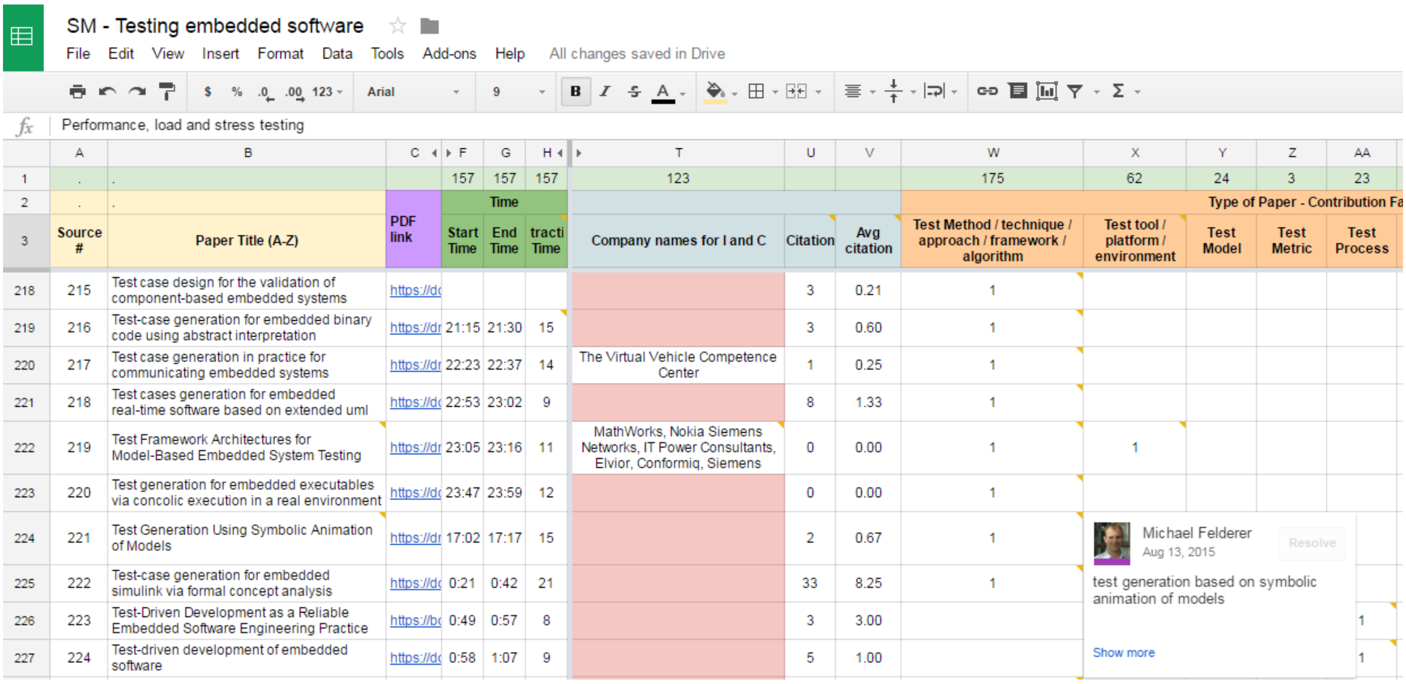
Group 2- Specific to the domain (testing embedded software)	2.2	Types of testing activities	{Test-case design (criteria-based), test-case design (human knowledge-based), test scripting, test execution, test evaluation (oracle), test automation, test management, other testing activities}	M
	2.3	Types of test artifacts generated	{Test case requirements (not input values), test case input (values), expected outputs (oracle), test code (e.g., in xUnit) and other}	M
	2.4	Types of non-functional testing, if any	{Performance, load and stress testing, real-time, reliability, other}	M
	2.5	Techniques to derive test artifacts	{Requirement-based testing (which includes model-based testing), code-coverage analysis, risk/fault-based testing, and search-based testing, other}	M
	2.6	Types of models used in model-based testing	{Finite state machine (FSM) and extensions, MATLAB Simulink models, other}	M
	2.7	Testing tools (used or proposed)	Name(s) of testing tool(s) used and/or proposed in the paper	M
	2.8	Types of evaluation method	{Example/applicability, coverage (code, model), detecting real faults, detecting artificial faults, mutation testing (fault injection), time/performance, other}	M
	2.9	Operating systems (OS)	As indicated in the paper	M
Group 3- Specific to system under testing (SUT)	3.1	Simulated or real system	{Simulated, real system}	S
	3.2	Number of SUTs (examples)	Integer value, as indicated in the paper	S
	3.3	Types of SUT (or example)	{Academic experimental (simple examples), real open-source, commercial systems}	M
	3.4	SUT programming language(s)	Programming language(s) as indicated in the paper	M
	3.5	SUT and board name(s)	SUT and board name(s) as indicated in the paper	M
	3.6	Application domains/industries	{Generic, home appliances and entertainment, aviation, avionics and space, automotive, defense, industrial automation /control, medical, mobile and telecom, transportation, other}	M
Group 5-Trends and demographics	4.1	Affiliation types of the study authors	{A: Academic, I: Industry, C: collaboration}	S
	4.2	Active companies	Name(s) of the company (ies) involved in the paper	M
	4.3	Highly cited papers	Citation count form Google Scholar, extracted on Feb. 28, 2016	

5.2 Data extraction and synthesis

Once the systematic map (classification scheme) was ready, each of the researchers extracted and analyzed data from the subset of the sources (assigned to her/him). We included traceability links on the extracted data to the exact phrases in the sources to ensure that how the classification is made is suitably justified.

Figure 5 shows a snapshot of our online spreadsheet that was used to enable collaborative work and classification of sources with traceability links (as comments). In this snapshot, classification of sources w.r.t. RQ 1.1 (Contribution type) is shown and one researcher has placed the exact phrase from the source as the traceability link to facilitate peer reviewing and also quality assurance of data extractions.

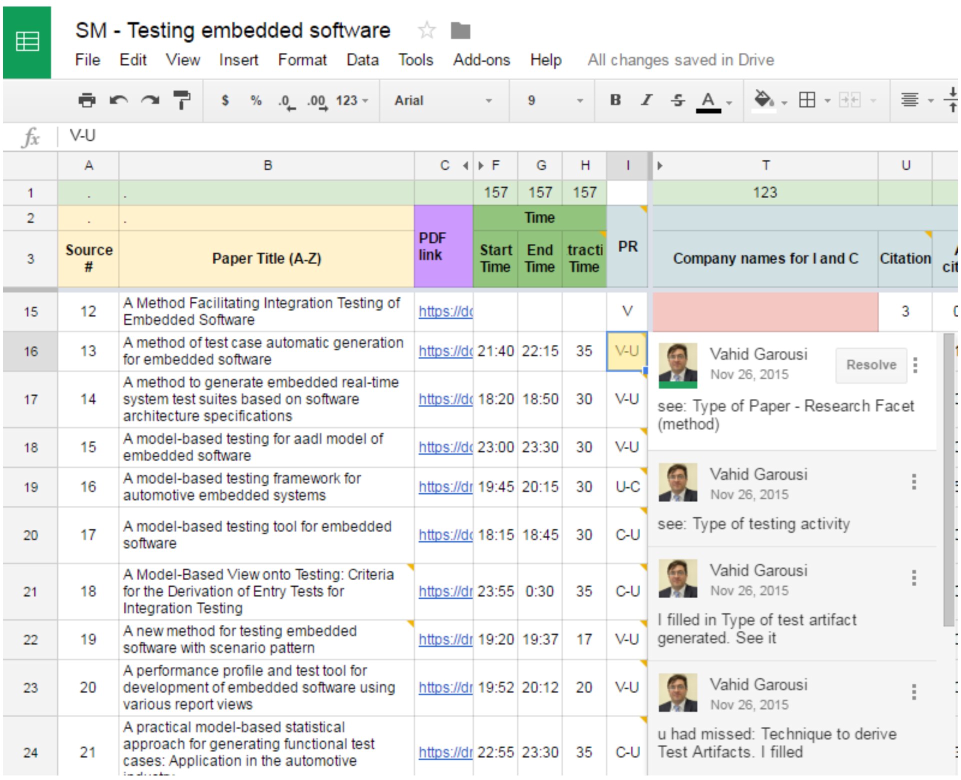
This is the pre-print of a paper that has been published: http://dx.doi.org/10.1016/j.infsof.2018.06.016


SM - Testing embedded software
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive
Performance, load and stress testing

	A	B	C	F	G	H	T	U	V	W	X	Y	Z	AA
1	.	.		157	157	157	123			175	62	24	3	23
2	.	.		Time						Type of Paper - Contribution Fa				
3	Source #	Paper Title (A-Z)	PDF link	Start Time	End Time	tracti Time	Company names for I and C	Citation	Avg citation	Test Method / technique / approach / framework / algorithm	Test tool / platform / environment	Test Model	Test Metric	Test Process
218	215	Test case design for the validation of component-based embedded systems	https://dc					3	0.21	1				
219	216	Test-case generation for embedded binary code using abstract interpretation	https://dr	21:15	21:30	15		3	0.60	1				
220	217	Test case generation in practice for communicating embedded systems	https://dr	22:23	22:37	14	The Virtual Vehicle Competence Center	1	0.25	1				
221	218	Test cases generation for embedded real-time software based on extended uml	https://dc	22:53	23:02	9		8	1.33	1				
222	219	Test Framework Architectures for Model-Based Embedded System Testing	https://dr	23:05	23:16	11	MathWorks, Nokia Siemens Networks, IT Power Consultants, Elvior, Conformiq, Siemens	0	0.00	1	1			
223	220	Test generation for embedded executables via concolic execution in a real environment	https://dc	23:47	23:59	12		0	0.00	1				
224	221	Test Generation Using Symbolic Animation of Models	https://dr	17:02	17:17	15		2	0.67	1				
225	222	Test-case generation for embedded simulink via formal concept analysis	https://dc	0:21	0:42	21		33	8.25	1				
226	223	Test-Driven Development as a Reliable Embedded Software Engineering Practice	https://bc	0:49	0:57	8		3	3.00					1
227	224	Test-driven development of embedded software	https://dc	0:58	1:07	9		5	1.00					1

Figure 5- A snapshot of the online spreadsheet that was used to enable collaborative work and classification of sources with traceability link to the primary studies (an example is shown)

After all researchers finished data extractions, we conducted systematic peer reviewing in which researchers peer reviewed the results of each other's analyses and extractions. In the case of disagreements, discussions were conducted. This was conducted to ensure quality and validity of our results. Figure 6 shows a snapshot of how the systematic peer reviewing was done.


SM - Testing embedded software
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive
V-U

	A	B	C	F	G	H	I	T	U
1	.	.		157	157	157		123	
2	.	.		Time					
3	Source #	Paper Title (A-Z)	PDF link	Start Time	End Time	tracti Time	PR	Company names for I and C	Citation
15	12	A Method Facilitating Integration Testing of Embedded Software	https://dc				V		3
16	13	A method of test case automatic generation for embedded software	https://dc	21:40	22:15	35	V-U		
17	14	A method to generate embedded real-time system test suites based on software architecture specifications	https://dc	18:20	18:50	30	V-U		
18	15	A model-based testing for aadl model of embedded software	https://dc	23:00	23:30	30	V-U		
19	16	A model-based testing framework for automotive embedded systems	https://dr	19:45	20:15	30	U-C		
20	17	A model-based testing tool for embedded software	https://dc	18:15	18:45	30	C-U		
21	18	A Model-Based View onto Testing: Criteria for the Derivation of Entry Tests for Integration Testing	https://dr	23:55	0:30	35	C-U		
22	19	A new method for testing embedded software with scenario pattern	https://dr	19:20	19:37	17	V-U		
23	20	A performance profile and test tool for development of embedded software using various report views	https://dr	19:52	20:12	20	V-U		
24	21	A practical model-based statistical approach for generating functional test cases: Application in the automotive	https://dr	22:55	23:30	35	C-U		

Figure 6- A snapshot showing how the systematic peer reviewing was orchestrated and conducted

6 Results

Results of the systematic mapping are presented in this section from Section 6.1 to 6.4.

6.1 Group 1-Contribution and research facets

We address RQ 1.1- RQ 1.2 in this section.

6.1.1 RQ 1.1: Mapping of studies by contribution facet

Figure 7 shows the cumulative trend of mapping of studies by contribution facet. Until the end of the review period (year 2017), out of the 312 sources in the pool, 204 (57.7% of the pool) presented test methods/techniques. A review of those techniques will be presented in Section 6.2.5 (RQ 2.5-Technique to derive test artifacts).

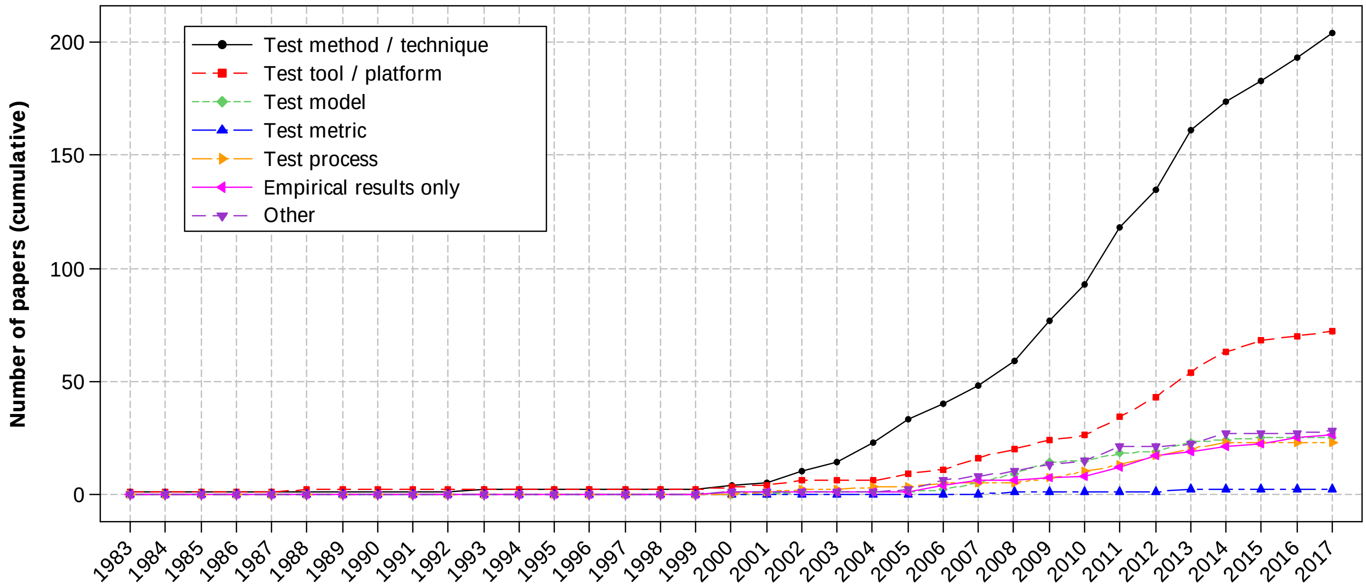


Figure 7-Cumulative trend of mapping of studies by contribution facet

Out of the 312 sources in the pool, 72 papers (21.2% of the pool) contributed test tools or platforms. A review of those techniques will be presented in Section 6.2.7 (RQ 2.7-Testing tools).

25 papers (7.7% of the pool) presented test models to assist test activities. For example, [Source 36] presented the *Embedded Test Process Improvement Model (Emb-TPI)* to conducted test process improvement in the context of embedded systems. As another example, [Source 58] proposed a test model to test hardware interfaces and OS interfaces for embedded systems.

2 papers (0.6%) contributed test metrics to support test activities. For example, [Source 133] presented a metric for measuring embedded software testability. [Source 207] presented a specific coverage metric for embedded software called 'variants coverage'.

23 papers (7.4%) presented test processes specific for embedded software. For example, [Source 41] presented a process to develop adaptive object-oriented scenario-based test frameworks for testing embedded systems. [Source 72] presented a statistical testing process for testing embedded systems which involves the following six steps: usage model construction, model analysis and validation, tool chain development, test planning, testing, and product and process measurement.

The contribution of 26 papers (6.7%) were empirical studies and empirical results. For example, [Source 1] presented a case study of black-box testing for embedded software using a specific test automation tool. [Source 9] presented an experimental evaluation of automated test input generation in the Java platform testing in the context of a specific embedded device. [Source 165] presented an empirical study on model-based testing of configurable embedded systems in the automation domain.

28 papers (8.7%) presented “Other” types of contributions. For example, [Source 35] presented a taxonomy of model-based testing for embedded systems which was generated from multiple industry domains. Several different test architectures for testing embedded software were presented in [Sources 46, 77, 118]. Entitled ‘Effective test driven development for embedded software’, [Source 124] presented a test pattern called “model-conductor-hardware”. [Source 126] presented a fault model specific for testing embedded systems. [Source 192] presented a set of mutation operators for mutation testing of embedded software. [Source 198] presented a specific test-scripting language.

We also wanted to get an overview of the topics covered in the papers according to their titles. Word clouds are a suitable tool for this purpose. Figure 17 shows a word cloud of all paper titles denoting the popularity of the topics covered (we used the online tool www.wordle.net). As we can see in this bird’s eye view, topics such as model-based and automated/automatic (testing), (test-case) generation, and control systems are among the most popular topics.

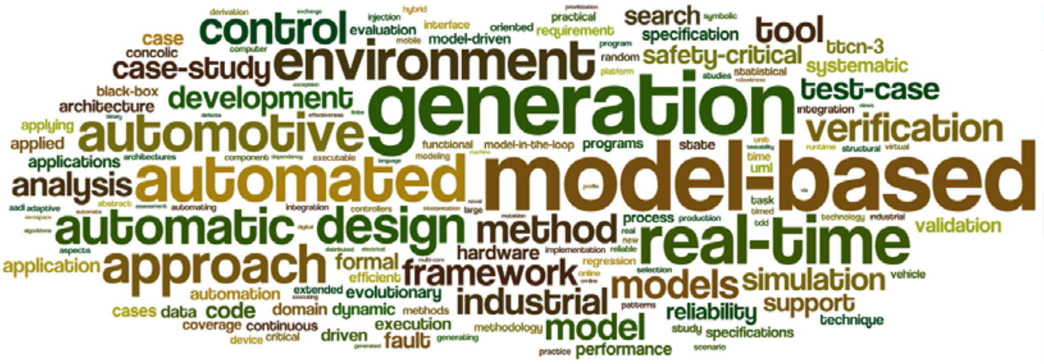


Figure 8-Popularity of the topics shown by the word cloud of all paper titles

6.1.2 RQ 1.2: Mapping of studies by research facet

Figure 9 shows the cumulative trend of mapping of studies by research facet. As we can see, a large portion of papers (137 of 312, 43.9%) present solution proposals (by examples) without rigorous empirical studies. 98 (31.4%) papers are weak empirical studies (validation research). 36 (11.5%) are experience papers. 34 are strong empirical studies (evaluation research). 2 and 5 papers, respectively, are philosophical and opinion papers.

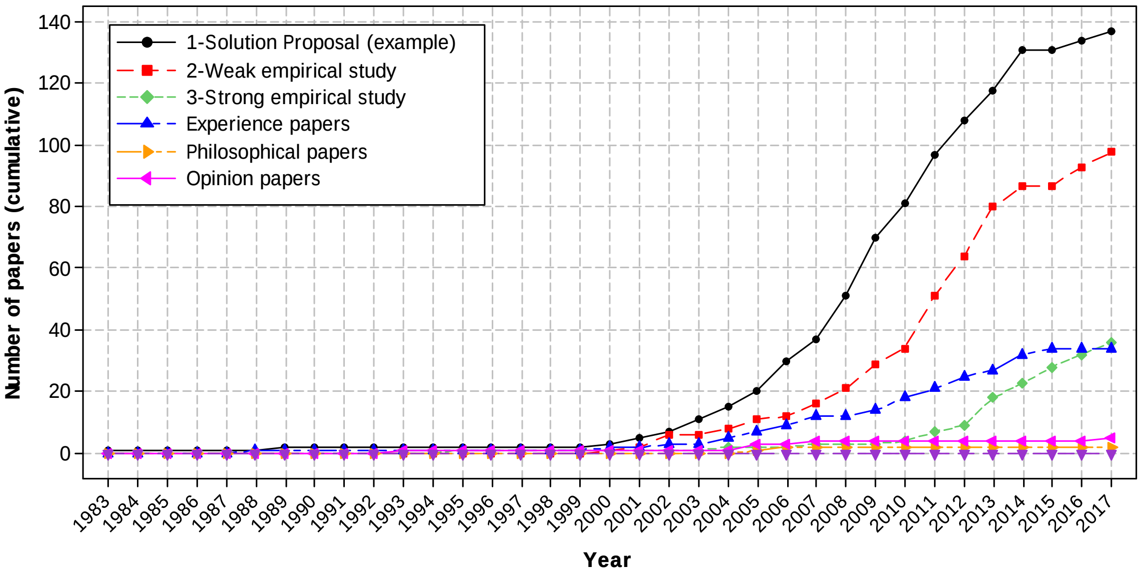


Figure 9-Cumulative trend of mapping of studies by research facet

Since "strong" empirical studies are the most rigorous studies in this context, we provide a few examples of those sources. Entitled "An approach to testing commercial embedded systems", [Source 45] presented a test adequacy criteria based on data-flow analysis and then an empirical study with two research questions:

- RQ1: Do black-box test suites augmented to achieve coverage in accordance with our first two adequacy criteria achieve better fault-detection effectiveness than test suites not so augmented, and if so to what extent?
- RQ2: Do test suites that are coverage-adequate in accordance with our first two adequacy criteria achieve better fault-detection effectiveness than equivalently-sized randomly generated test suites, and if so to what extent?

[Source 58] presented an interface test model for hardware-dependent software and API of embedded systems and then a comprehensive empirical study including careful measurement of frequency of interface faults of fault detecting capability. [Source 68] presented a search-based approach for automated model-in-the-loop testing of continuous controllers and then a comprehensive empirical study with several RQs.

Entitled "Automated system testing of real-time embedded systems based on environment models ", [Source 73] raised and addressed the following three research questions:

- RQ1: What is the effect of test case representation on fault detection effectiveness of the testing strategies?
- RQ2: Which testing strategy is best in terms of failure detection?
- RQ3: Is environment model-based system testing an effective approach in detecting faults for industrial embedded systems?

6.2 Group 2-Specific to the domain (testing embedded software)

We address RQ 2.1- RQ 2.9 in this section.

6.2.1 RQ 2.1-Level of testing

In terms of level of testing considered in the papers, most of them (233 papers) considered system testing. 89 and 36 papers, respectively, focused on unit and integration testing. The detailed classification of papers can be found in the online spreadsheet (goo.gl/MhtbLD).

By focusing on unit testing, [Source 63] applied test-driven development (TDD) to embedded software. Several practical examples of automated unit test code were provided.

In [Source 99], a case study of combinatorial testing for an automotive hybrid electric vehicle control system was reported. The combinatorial test approach was applied to a real Hybrid Electric Vehicle control system as part of a hardware-in-the-loop test system. The paper was thus classified under system testing. In the test approach presented in [Source 54], integration testing was performed using hardware-in-the-loop approach.

6.2.2 RQ 2.2-Types of test activities

Inspired by books such as [56] and our recent SLM and SLR studies on software testing such as [45, 46], we categorized testing activities based on the generic test process shown in Figure 10. Testing usually starts with test-case design (either criteria-based or human knowledge-based). Using the derived test cases, test execution follows afterwards in which the System Under Test (SUT) is tested (exercised). Test evaluation (using test oracles) is the final phase in which the results of testing are evaluated (pass or fail), and test verdicts are made. Three cross-cutting activities are also shown in Figure 10: test management (planning, control, monitoring, etc.), test automation (could be conducted in any of the phases), and other activities (e.g., regression testing, and test prioritization). Note that many people think of test automation only for automated execution of test cases, but test automation has been successfully implemented in other test activities too, e.g., test-case design and test evaluation.

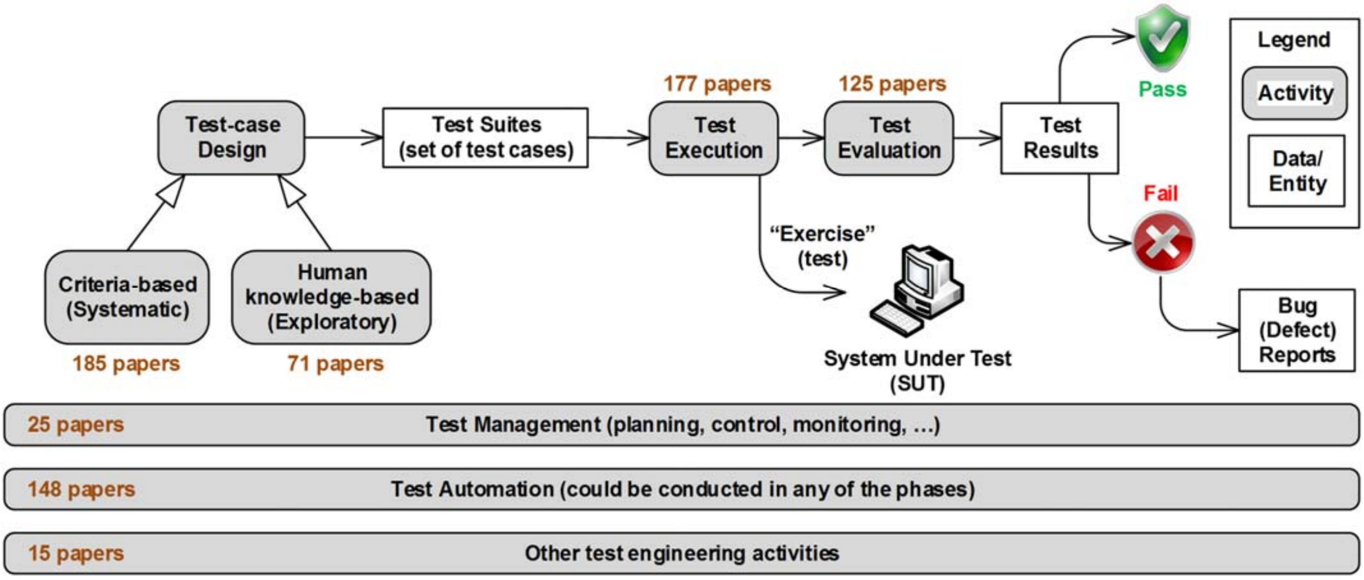


Figure 10- A generic test process showing different types of test activities

As we can see in Figure 10, there is a good mix of papers proposing techniques and tools for each of the test activities. We can notice the major focus on test execution, automation and criteria-based test-case design (e.g., based on code coverage). The sum of the numbers in Figure 10 are more than the number of papers (312), since many papers made contributions in more than one test activity, e.g., the paper "*Improving the accuracy of automated GUI testing for embedded systems*" [Source 155] made contributions to four test activities: human knowledge-based test-case design, test automation, test execution and test evaluation.

To know about the existing advances and to potentially adopt/customize them for usage in their own testing needs, we advise test practitioners in the embedded software industry to review the list of 312 studies in our online spreadsheet (goo.gl/MhtbLD) as categorized by the above six types of test activities. For example, if a test team intends to conduct test-case design, it is advised to review the 185 papers in the "criteria-based" category or the 71 papers in the "human knowledge-based" test-case design category to see if the existing approaches can be adopted/customized into their needs. This prevents them from "reinventing the wheel" (developing an already-existing test technique). We review a few example sources below.

Test-case design (criteria-based):

[Source 17] presented a method to generate embedded real-time system test suites based on software architecture specifications. The method maps specifications in a specific description language named DRTSADL into a format of timed input/output automaton. In [Source 34], test sequences are generated based on Extended Finite State Machines (EFSM).

Test-case design (human knowledge-based):

In [Source 55], an industrial case study of structural testing applied to safety-critical embedded software was reported. In that work, manual functional tests were created by a test engineer at the company under study. They were created by hand, following a design validation test plan.

In [Source 68], a search-based approach for automated model-in-the-loop testing of continuous controllers was reported in which, based on domain expert knowledge, the authors selected the data regions that were more likely to include critical and realistic defects. That was thus a human knowledge-based test-design approach.

Test execution:

Many papers (177 of 312) had the test execution component in them, in addition to other test activities. In [Source 86], a tool named CoCoTest for model-in-the-loop (MiL) testing of continuous controllers was presented. [Source 133] presented hardware-in-the-loop search–based testing approach and a tool named MESSINA for the execution of hardware and software test sequences.

Test automation:

Test automation is a very popular approach to reduce testing costs for testing different types of software including embedded software [55, 70]. 148 of the 312 papers involved test automation.

In [Source 42], an automated approach to reducing test suites for testing embedded systems was presented, in which a test suite generator automatically generates a test suite using the C grammar. The authors of [Source 57] presented an automated test case generator tool that uses Genetic algorithms (GAs) to automate the generation of test cases from output domain and the critical regions of an embedded System.

Test evaluation (using test oracles):

Test evaluation (using test oracles) is also popular in this area as 125 of the 312 papers considered it. In [Source 15], a model-based testing approach to generate test cases and oracles based on the Architecture Analysis & Design Language (AADL) was presented. The presented tool can generate the test input pool and testing oracles according to the AADL specifications.

In [Source 50] too, a test tool automates test item identification, test case generation, and determination of 'pass or fail' in runtime environment.

Test management:

Only a small number of papers (25 of 312) addressed test management in the context of embedded systems. [Source 36] addressed test process improvement.

Entitled "*Formal specification and systematic model-driven testing of embedded automotive systems*", [Source 148] proposed guidelines for test planning using a set of models.

In [Source 168], model-driven testing of embedded automotive systems with timed usage models was discussed, in which the usage models serve as the basis for the whole testing process, including test planning and test-case generation.

Other test activities:

15 of the 312 papers focused on other test activities. In [Source 108], an approach for test-case minimization and prioritization specific for embedded systems was presented. [Source 159] and [Source 183] presented approaches for test reuse.

Entitled "*Rapid embedded system testing using verification patterns*", [Source 208] presented a set of practical test patterns.

6.2.3 RQ 2.3-Types of test artifacts generated

Different test techniques have been proposed to generate different types of test artifacts. Ordered by frequency (number of papers), the largest ratio of papers (144 of 312) proposed techniques to derive test case inputs (values), e.g., the paper "*Applying model-based testing in the telecommunication domain*" [Source 60] applied model-based coverage criteria to derive test cases.

In 100 papers, approaches for generation of test case requirements (not explicit input values) are presented. As discussed above, test requirements are usually not actual test input values, but the conditions (e.g., for control flow paths in code) that can be used to generate test inputs.

In 95 papers, the generation of automated test code (e.g., in xUnit) is addressed. For example, the paper "*A model-based testing framework for automotive embedded systems*" [Source 19] developed an approach to generate test scripts in Python based on a specific form of abstract test-cases.

In 80 papers, expected outputs (test oracle) or their generation are discussed. For example, [Source 19] proposed an approach for generating expected outputs using the specifications based on the Architecture Analysis and Design Language

(AADL). 15 papers proposed "Other" types of test artifacts, e.g., test patterns in [Source 83] and test documentation in [Source 163].

6.2.4 RQ 2.4-Type of non-functional testing, if any

As discussed in Section 3.2, while our focus was only to include functional testing papers, but some of the functional testing papers also discussed non-functional testing aspects as well, e.g., security and load testing. We extracted those information as well. 25 papers conducted performance and load (stress) testing. 17 and 10 papers, respectively, conducted real-time and reliability testing. 12 papers cover other types of non-functional testing including security.

In [Source 33], a XML-based testing tool for profiling and testing embedded software's performance was presented. [Source 59] presented a search-based technique to generate stress test cases attempting to violate performance requirements

[Source 200] presented an industrial field study of software-generated device exception for intensive device-related software testing in Hyundai Motor Corporation. The proposed method is a reliability testing technique to verify whether software components that are connected to a hardware device properly handle errors caused by the hardware.

6.2.5 RQ 2.5-Techniques to derive test artifacts

As discussed in our classification scheme (Table 3), we categorized the techniques to derive test artifacts into six groups: (1) Requirement-based testing, includes model-based testing, (2) White-box testing (code-coverage analysis), (3) Risk/fault-based testing, (4) Search-based testing, (5) Random testing, and (6) Other techniques.

180 papers used requirement-based testing (including model-based testing). For example, in [Source 16], a model-based testing framework for automotive embedded systems was presented in which the methodology relies on: automated model-based test-case generation for functional requirements criteria based on the models created using the Architecture Description Language (ADL) extended with timed automata semantics.

53 papers used white-box testing (code-coverage analysis) to derive test artifacts. For example, in [Source 43], an automated testing experiment for layered embedded C code was reported in which test cases were automatically generated using test criteria such as reachable state coverage and transition coverage.

24 papers used risk/fault-based testing to derive test artifacts. An example of the papers under the risk/fault-based testing group is [Source 168] in which mutants were generated for embedded systems using kernel-based software and hardware fault simulation.

24 papers used search-based testing to derive test artifacts. Entitled "*A highly configurable test system for evolutionary black-box testing of embedded systems*", [Source 9] presented a tool that allows full automation of black-box tests on different testing platforms (MiL, SiL, HiL) by applying search-based testing techniques.

23 papers used random testing. In several papers, e.g., [Source 37] and [Source 40], test-cases generated using systematic approaches (e.g., requirement-based testing) were compared with those generated from random testing.

24 papers used other techniques to derive test artifacts. Techniques other than the above ones were also developed and used to derive test artifacts, e.g., [Source 54] used a technique called "orthogonal array-based robust testing (OART)". Entitled "*Automated generation of test trajectories for embedded flight control systems*", [Source 58] defined a set of formal regressive models to derive regression test cases for testing embedded flight control systems. Concolic (a portmanteau of concrete and symbolic) testing was used in [Source 90] and was applied in several case studies on mobile programs. Cleanroom statistical testing was applied in [Source 248].

For a practitioner who is interested to adopt some of the presented techniques, it is important to know the advantages and disadvantages of using different techniques, whether these techniques are practical, and how much manual effort is involved in applying them. Such critical and "comparative" analysis are quite rare, i.e., only one paper in the pool [Source 3] reported a comparative study of manual and automated testing in industrial control software. The results of that study, [Source 3], showed that automatically-generated test suites achieve similar code coverage as manually-created test suites, but in a fraction of the time (an average improvement of roughly 90%). We also found that the use of an automated test generation tool does not necessarily result in better fault detection in terms of mutation score compared to manual testing. Specifically, the empirical study found that manual test suites more effectively detect logical, timer and negation type of faults, compared to automatically-generated tests.

6.2.6 RQ 2.6-Types of models used in model-based testing

As discussed above, in terms of techniques used to derive test artifacts, requirement-based testing is very popular in this area as it was used in 180 papers (%57.6 of the papers in the pool). A large portion of these papers (150 of 180) used model-based testing (%48.0 of the pool).

Figure 5 shows the general process of model-based testing, in which models are developed using either a forward engineering or a backward engineering manner. Once models are validated themselves, they can be used for test-case generation (test-case design), e.g., Finite State Machines (FSM) and its extensions are frequently used to derive test-case sequences using coverage criteria such as all-transitions coverage. Given the large wealth of knowledge and industrial evidence in model-based testing of embedded systems (150 papers), we recommend companies, who are planning to implement systematic testing, to review and consider this vast body of knowledge to potentially adopt some of the ideas in model-based testing.

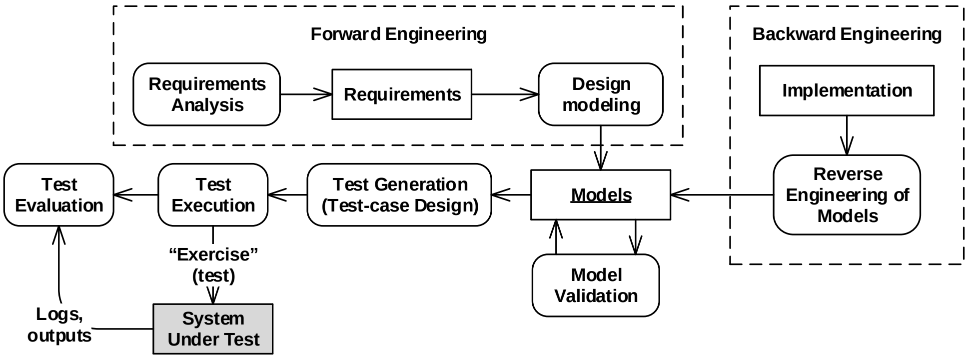


Figure 11- General process of model-based testing

In 50 papers, Finite State Machines (FSM) and its extensions (e.g., timed-FSM) were used for model-based testing, e.g., [Sources 16, 18, 21]. 32 papers used MATLAB Simulink models, e.g., [Sources 28, 29, 54]. 85 papers used "Other" types of models, e.g., Method Definition Language (MeDeLa) in [Source 170] and UML sequence diagrams in [Source 173].

Model-based testing actually fits in the scope of a larger development concept for embedded systems, i.e., X-in-the-loop development, simulation and testing [Sources 160, 161, 194, 208]: which consist of Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), Hardware-in-the-Loop (HiL), and System-in-the-Loop (SYSiL) as shown in Table 4. X-in-the-loop testing [71] has gained acceptance due to the increased adoption of model-based development (MBD) in industry, especially in the automotive domain [72]. MBD and in-the-loop methods are widely used today in automotive to develop and test control systems. A complete in-the-loop flow is composed of several chronological steps as illustrated in Figure 12. First, on the left side of the V cycle, MiL is applied, followed by SiL. On the right side of the V cycle, PiL is applied after SiL. Then, HiL and SYSiL are applied.

MiL testing is the simulation and testing of an embedded system in an early development stage, the modeling, in model-based software development. Embedded systems interact with their environment and often expect implausible sensor signals as input and then stimulate the physical system. To function properly, the environment of the embedded system must be simulated. Since it can be applied in early development stages, MiL is an inexpensive way to test embedded systems. As discussed above, a major portion of MiL testing in industry is conducted using MATLAB Simulink models, e.g., [Sources 28, 29, 54].

Once the model is verified and tested using MiL, the next stage is SiL where the embedded software engineers develop the software code depending on the processor or FPGA that is planned to be used in the final hardware implementation and then they execute the tests and simulations for the controller model on the PC platform (with the system, also called "plant", still as a software model) with this code to verify it. If any glitches are detected, engineers can go back to MiL and make necessary changes.

The next test phase is the PiL which goes beyond just the PC platform (SiL). This step introduces some hardware features that permit to achieve more realistic situations where the control algorithm will run. In PiL, a target (external) processor is chosen and the communication with the external processors is conducted by using specific functions installed in a simulation integrated environment installed on the host PC.

The next phase is HiL which provides an effective platform by adding the complexity of the "plant" (system) under control to the test platform, by adding a mathematical representation of all related dynamic systems. These mathematical representations are referred to as the "plant simulation". The embedded system to be tested interacts with this plant simulation. The very last phase is System-in-the-Loop (SYSiL) testing in which the actual physical embedded system is tested.

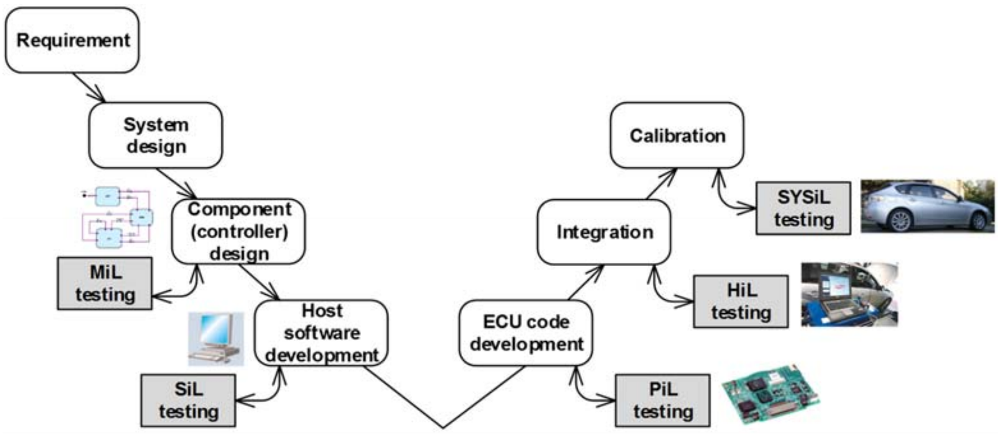


Figure 12- The "V" development process including the X-in-the-loop testing phases (adapted from [72])

Table 4-Diffeent types of X-in-the-loop testing phases

Phase/level	**Entity under test**	**Testing interfaces**
Model-in-the-Loop (MiL) testing	System model	Messages and events of the model
Software-in-the-Loop (SiL) testing	Control software (e.g., C or Java code)	Methods, procedures, parameters, and variables of the software
Processor-in-the-Loop (PiL) testing	Binary code on a host machine emulating the behavior of the target	Register values and memory contents of the emulator
Hardware-in-the-Loop (HiL) testing	Binary code on the target architecture	I/O pins of the target microcontroller or board
System-in-the-Loop (SYSiL) testing	Actual physical embedded system	Physical interfaces, buttons, switches, displays, etc.

6.2.7 RQ 2.7-Testing tools (used or proposed)

106 papers used existing test tools and 98 papers proposed new test tools.

In [Source 63], "*Applying test driven development to embedded software*", the CppUnit framework was used. In [Source 77], NUnit was used. Tools Sikuli and MonkeyRunner were used in [Source 155]. In [Source 159], test cases were implemented using MATLAB Simulink and Stateflow tools.

In [Source 106], E-TDD (an embedded test-driven development tool) was proposed. [Source 159] proposed a tool called MiLEST for Model in-the-Loop testing. In [Source 186], "*Motocap STA framework: new approach to testing of embedded systems*", a tool named Motorola OCAP (OpenCable Applications Platform) was proposed.

While we discuss above only a small subset of the tools, the names of all the used and proposed testing tools is available in the online spreadsheet (goo.gl/MhtbLD).

6.2.8 RQ 2.8-Type of evaluation methods

As discussed in our classification scheme (Table 3), we categorized the evaluation methods conducted in the papers into these categories (we also specify the number of pears in each category): (1) Examples showing the applicability of test approaches (167 papers), (2) coverage of code or model (74 papers), (3) detecting real faults (37 papers), (4) detecting artificial faults, (mutation testing, fault injection) (54 papers), (5) time/performance of test approaches (60 papers), and (6) other evaluation methods (26 papers). The first method (examples showing the applicability) is the most trivial one. We discuss a few examples under the 'other' category next.

In [Source 76], "*Automated testing of embedded automotive systems from requirement specification models*", it is argued that the fulfillment of requirements of international standards such as the ISO-26262 is facilitated with the presented approach. In [Source 77], to evaluate the tool presented, a functionality comparison assessment was conducted between the presented system called ATEMES and the relevant tools. In [Source 180], *"Model-based testing of highly configurable embedded systems in the automation domain"*, the ratio of test case 'reuse' during a large test-case development project was measured.

In [Source 180], a case study for verification of an implantable medical device was conducted and testing productivity using the proposed approach was measured.

6.2.9 RQ 2.9-Operating systems (OS)

Some of the papers explicitly reported the operating systems in which the approaches were implemented and the experiments were conducted. In terms of majority, the most common OS's were: Unix/Linux (in 23 papers), vxWorks (in 5 papers), the Windows family (in 5 papers), and other OS's (in 19 papers). The others included: QNX in [Source 22], MicroC/OS-II in [Source 40] and [Source 50], Symbian OS in [Source 177], Embedded Configurable Operating System (eCos) in [Source 240], OORTX-RXF in [Source 255], and uC/OS-II in [Source 252].

6.3 Group 3-Specific to system under testing (SUT)

We address RQ 3.1- RQ 3.6 in this section.

6.3.1 RQ 3.1-Simulated or real systems

Since development and testing of embedded systems in real environments is not always easy or practical (e.g., the control software of a fighter jet), development and testing of those systems in simulated environments first (before real systems) are common. A popular approach in this context is X-in-the-loop development, simulation and testing [71]: which consist of Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), and Hardware-in-the-Loop (HiL).

Figure 13 shows the breakdown of the number of papers in terms of using simulated or real SUTs in the evaluations. As we can see, testing with the real systems in place is the most common. After that, MiL, SiL and HiL are the next common approaches in order.

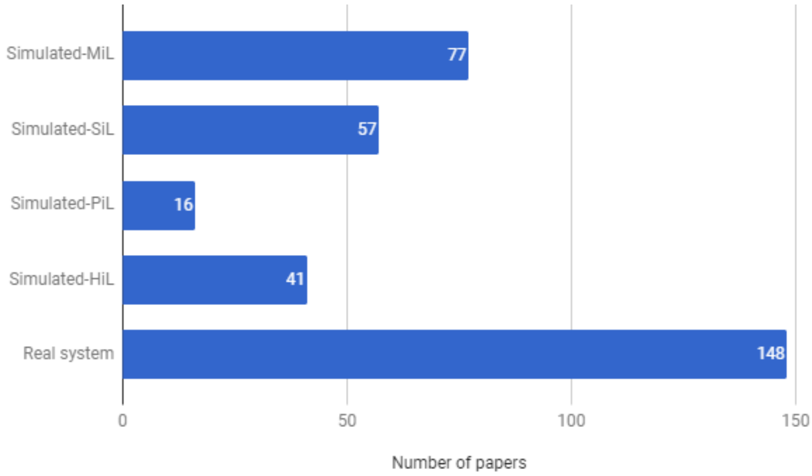


Figure 13- Breakdown of the number of papers in terms of using simulated or real SUTs in the evaluations

In [Source 9], the authors followed the MiL testing approach and mentioned that "*the ABS system [model] had to be created to simulate the system environment of the ABS system in such a manner that the anti-lock braking system detects no difference to its use in a vehicle*". In [Source 21], SiL testing was conducted as test cases were executed in a simulated environment (host PC). MiL was also used in [Source 28] as "*the executable model of the embedded system formed the focal point of the model-based test strategy*".

6.3.2 RQ 3.2-Number of SUTs (examples)

One would expect that each paper applies the proposed testing technique to at least one SUT. Some papers take a more comprehensive approach and apply the proposed testing technique to more SUTs. Figure 14 shows the histogram of the number of SUTs (examples) in the papers. As we can see, most papers (197 of them) applied the approaches to one SUT (or example) only.

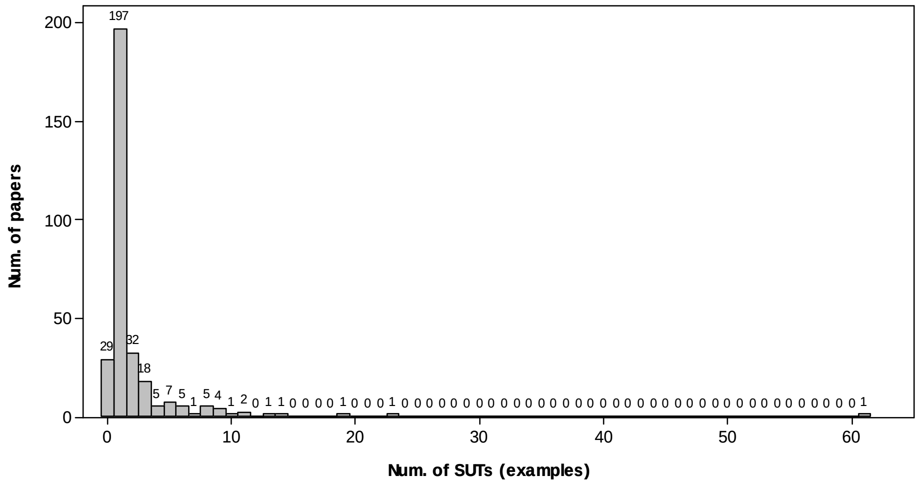


Figure 14- Histogram of the number of SUTs (examples) in the papers

The comprehensive study in this aspect was [Source 27, 193]. In [Source 27], a profiling method for memory fault detection was reported and was evaluated in an industrial field study on 23 systems. In [Source 193], mutation-based test generation for PLC embedded software was conducted on 61 programs provided by Bombardier Transportation company.

6.3.3 RQ 3.3-Type/scale of SUTs (or examples)

The SUTs in some papers were academic experimental or simple examples, while the systems in other papers were real open-source or commercial systems. Almost equal number of papers (127 and 137) experimented their proposed ideas with academic experimental and real commercial systems. A few studies (12 papers) experimented their proposed ideas on real open-source embedded software.

Some of the real commercial systems under test in the papers were the followings. A real controller system for a car's exterior mirror was tested in [Source 39]. A set of smart card were tested in [Source 43]. An automated target tracking radar system (named ATTR) was under test in [Source 54]. A home security system was considered in [Source 55]. A vehicle adaptive cruise controller system made by Volvo was tested in [Source 131].

Furthermore, a type of embedded software that provides the low-level control for a system specific hardware is called *firmware* or *device driver*. Some papers used the "firmware"/"device driver" terminologies, e.g., in [Source 132], faults were injected in the SUT, which was a safety-critical automotive firmware. The SUT in [Source 20] was a Linux device driver, and one of the SUTs in [Source 78] was a flash memory device driver.

6.3.4 RQ 3.4-SUT programming/development languages

180 papers specified explicitly the programming/development languages of the SUTs. Figure 15 shows the histogram for this classification. As we can see and as we were expecting, C is the most popular programming language. MATLAB/Simulink, C++, and Java are the next popular ones in order. Examples of the "Other" programming languages are: Assembly in [Sources 26, 165, 188], C# in [Sources 69, 130], Lustre in [Source 78], SystemVerilog in [Source 131], HDL in [Source 153], and Verilog in [Source 214].

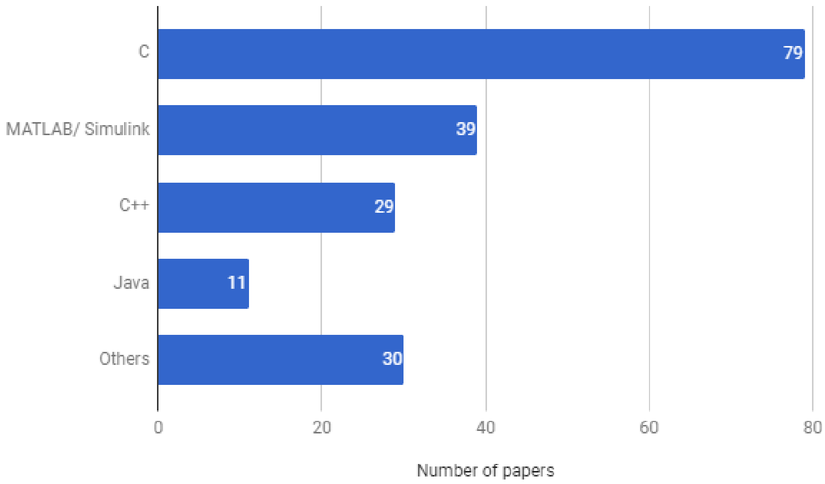


Figure 15- Histogram of the SUT programming/development languages in the papers

6.3.5 RQ 3.5-SUT and board name

216 and 35 papers, respectively, specified explicitly the names of the SUTs and hardware boards that they used. Table 5 lists the names of some of the interesting SUTs used in the studies. One can see that there is a good variety in terms of types and domains of these systems, e.g., from temperature controller through flight warning system to wireless metering system.

Table 5- Some of the interesting SUT names mentioned in the studies.

SUT name	Source #
Temperature controller	1
An embedded system of Mitsubishi Space Software Co. Ltd.	2
An industrial antilock brake system	3
Flight warning system	14
False coin inspector embedded software	23

Front wiper, and fuel gauge systems	26
Power window controller, cruise controller, and climate controller systems	28
Automotive air compressor module	59
Elevation control system	64
Movement control system of an Unmanned Aerial Vehicle (UAV)	67
A kitchen toaster	71
Window wiper control	73
Air conditioner controller	78
Hybrid electric vehicle control	88
The traffic light system	93
Sorting machine system, marine seismic acquisition system, gate system	120
High-speed excavator controller	125
Remote sensing in flight control system	129
Shuttle Remote Manipulator System (Robotic Arm)	141
Intelligent pick and place machine	153
Fuel level management system	154
Automatic transmission controller	156
Avionic embedded inertial navigation system (INS)	172
Fire alarm system	178
Defibrillator and car alarm system	180
Anti-lock braking system (ABS)	194
Flight computer system of satellite launch vehicle	199
Car infotainment system	200
Wireless metering system	230
Railway signaling system	260
Fuel injection control system	270

A few papers also explicitly mentioned the board names/models that they have used for experiments. Several papers have used boards with different types of ARM processors [Sources 20, 22, 42, 70, 77, 115, 248, 252]. Also boards containing Intel processors are referenced several times [Sources 42, 50, 95, 132]. In addition, several other lesser-known boards are covered, for instance Leon Embedded Microprocessor [Source 104] or Altera MAX9320 FPGA [Source 166].

6.3.6 RQ 3.6-Application domains/industries

While some test techniques are generic in the sense that they can be applied to all types of embedded software, some test techniques are domain-specific. According to the categorization made earlier in the paper (Section 5.1), Figure 16 depicts the breakdown of the different domains/industries for which the papers have focused on. Note that we have classified papers in these categories explicitly based on what was mentioned in each paper, and we did not reply on our own implicit personal interpretations.

As Figure 16 shows, research in the automotive sector is the most active in domain this field (96 papers, 30.8%). Industrial automation and control domain is the second most active domain (40 papers, 12.8%). The combined category of aviation, avionics and space industries is in the third rank. Among the "Other" domains are: banking, public transport and e-government in [Source 43], and fire-safety systems in [Source 219].

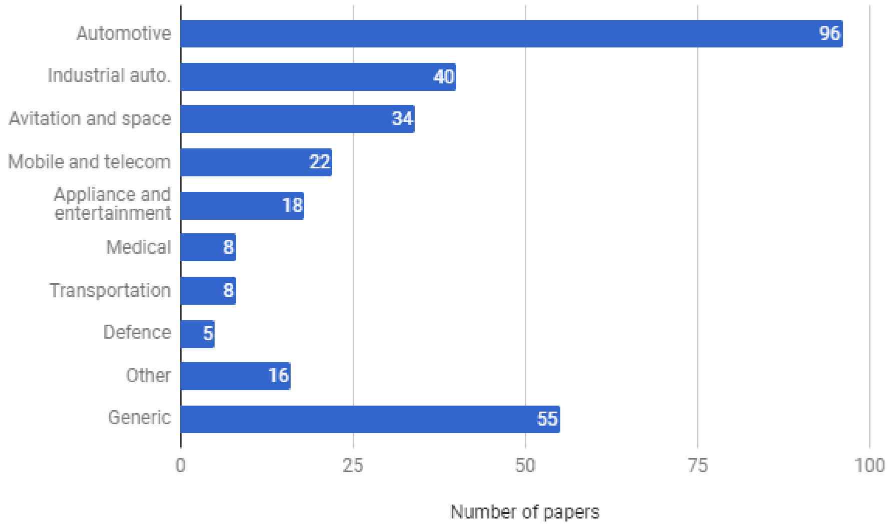


Figure 16- Breakdown of the different industries (domains) represented in the papers

6.4 Group 4-Demographic and bibliometric information

We address RQ 4.1- RQ 4.3 in this section.

6.4.1 RQ 4.1-Affiliation types of the study authors

We were curious to see the breakdown of the papers based on the affiliation types of the study authors. In Figure 17, we show (as a stack chart) the number of studies per year published solely by academic researchers (those working in universities and research centers/institutes), solely by practitioners (also including the researchers working in corporate research centers), or as collaborative work among academics and practitioners. As we can see, the attention level in this topic has steadily risen since early 2000's by both the research and practitioner communities. The peak year in terms of number of papers was year 2012 in which 38 papers were published.

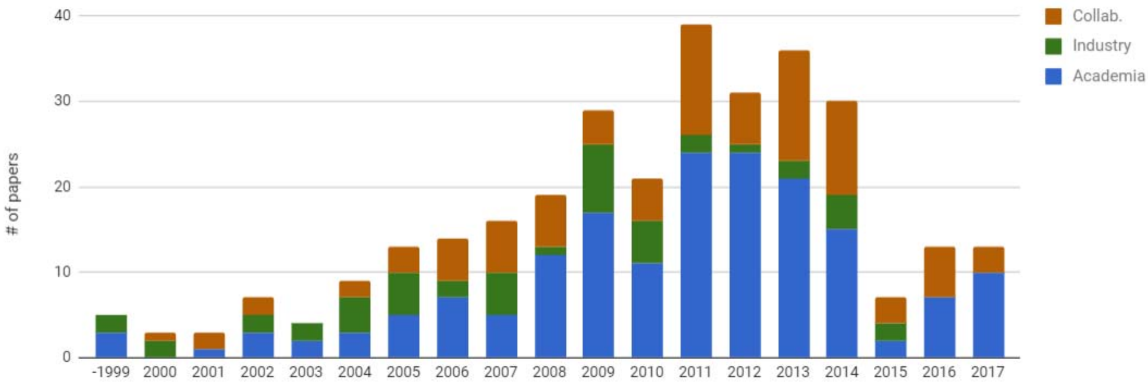


Figure 17-Growth of the field and affiliation types of the study authors across years

In terms of breakdown of papers by affiliation types of the authors, 172 papers (55.1%) were authored solely by academic researchers, 49 papers (15.7%) by practitioners only, and 91 papers (29.2%) as joint (collaborative) work between researchers and practitioners.

Papers solely by academic authors often cover formal methods, for instance [Sources 8, 71, 183], or other rather theoretical approaches like swarm optimization to test-case prioritization in the context of embedded systems [Source 53] or model-based simulation testing [Source 150]. Collaborative work comprises many industrial and practical case studies, for instance on statistical model-based testing [Source 21], combinatorial testing [Source 88], concolic testing [Source 136] and search-based testing [Source 192] as well as frameworks for model-based testing [Source 16], object-oriented testing [Source 36] or simulation testing [Source 254].

Finally, papers solely by industrial authors often cover practical issues, for instance a test process improvement model specific for embedded systems [Source 31], the application of test-driven development (TDD) [Source 55], test-data generation for C programs [Source 75], or test model processing tools [Source 236].

6.4.2 RQ 4.2-Active companies

In terms of the most active companies in conducting research and publishing papers in this area, the top three were: (1) Daimler (14 papers in the pool), (2) Samsung (8 papers), as well as (3) Nokia and Berner & Mattner (4 papers each).

Industrial and collaborative papers by Daimler cover a broad range from more theoretical to more practical topics including search-based testing [Source 24], model-based testing [Source 207], test-data generation [Source 74] as well as integration test levels [Source 138].

Collaborative papers by Samsung also range from more theoretical to more practical topics including concolic testing [Sources 70, 90, 136], test automation [Source 79] as well as test process improvement [Source 31].

Papers from Nokia cover multilevel testing for design verification [Source 166], aspect-based testing [Source 219] as well as distributed testing [Source 262]. Finally, papers from Berner & Mattner cover test systems for evolutionary black-box testing [Sources 3, 9] and search-based testing in the context of embedded systems [Source 194].

6.4.3 RQ 4.3-Citation analysis and highly-cited papers

The rationale behind this RQ is to characterize how the papers in this pool are cited by other papers, to get a sense of their impact and popularity, and also to identify the highly-cited papers (i.e., those with the highest impact) in the area so that readers can benefit from them. We did the citation analysis in a similar manner to our recent bibliometric studies, e.g., [57, 58, 73-75], in which we used two metrics: (1) the absolute number of citations to each paper, and (2) normalized citations (average number of citations per year). The citation data were extracted from Google Scholar on Feb. 28, 2016.

Using the above two metrics, Figure 18 shows the citation landscape of the 312 paper in the pool. 48 of the papers (17.6%) had no citations at all. The average values for the two metric values were 7.8, and 1.52, respectively, denoting that the papers in this area are reasonably cited. It is interesting to see that, in terms of normalized citations, more recent papers have higher citations.

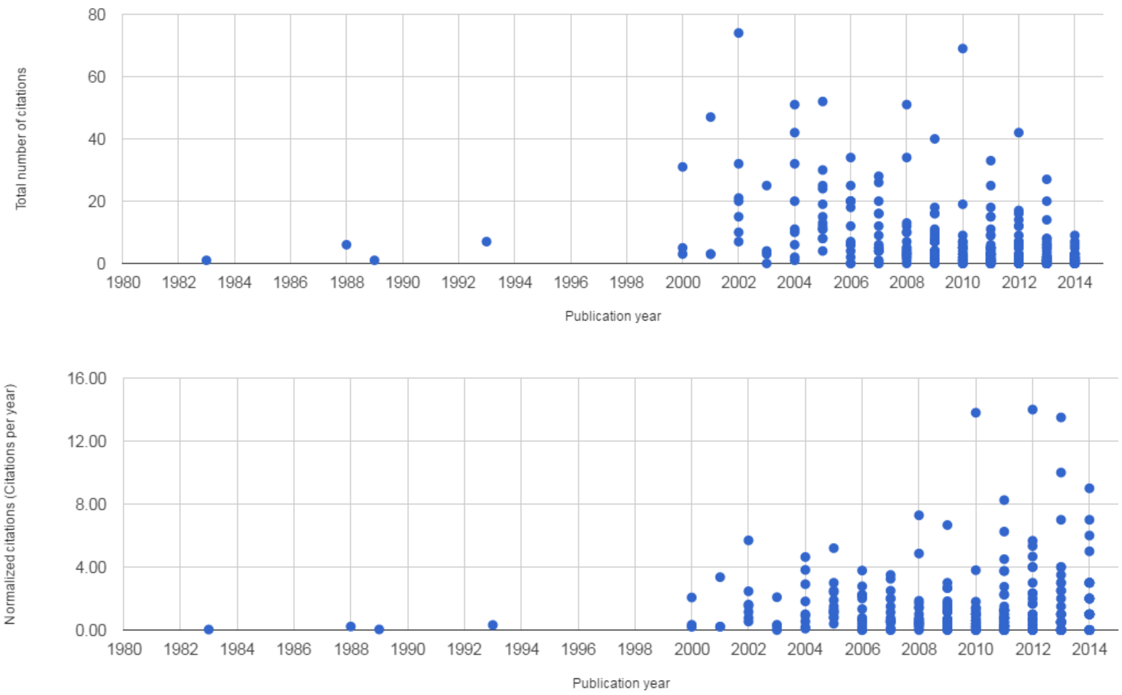


Figure 18- Citation analysis of the primary studies

The five most cited papers, based on the number of citations based on data gathered on date mentioned above, are listed in Table 6. Given their high impact and thus the quality of these papers, readers such as new researchers and graduate students are encouraged to read the highest cited papers and benefit from them.

Table 6- The five most cited papers in the pool based on the number of citations

Source #	Title	Year of publication	Num. of citations
86	Automatic test data generation for structural testing of embedded software systems by evolutionary testing	2002	96
35	A taxonomy of model-based testing for embedded systems from multiple industry domains	2006	87
60	Applying model-based testing in the telecommunication domain	2012	87
94	Black-box system testing of real-time embedded systems using random and search-based testing	2010	81
181	Model-based testing of real-time embedded systems in the automotive domain	2008	71

7 Discussion

In this section, we provide a summary of the findings, potential benefits of this review, limitations and potential threats to validity of our study.

7.1 Summary of the findings

We summarize the research findings of each RQ next.

Group 1-Common to all SLM studies:

- **RQ 1.1-Mapping of studies by contribution facet:** We saw that there is a mix of contribution types in different papers. The most common type of contribution are test tools or platforms, which are contributed by 63 papers. 24 papers present test models to assist test activities. 21 papers contribute empirical studies and empirical results. 2 papers provide test metrics to support test activities. Finally, 27 papers present other types of contributions, for instance a taxonomy for model-based testing or test patterns.

- **RQ 1.2-Mapping of studies by research facet:** The dominating research facet are solution proposals, which are presented in 126 papers (46% of the pool). These papers do not provide rigorous empirical studies and indicate the need for more empirical studies providing evidence on embedded software testing approaches. The second most common type contributed by 86 papers is validation research. These papers investigate properties of a solution that has not yet been implemented in practice. 32 papers are experience papers and only 22 provide evaluation research presenting strong empirical studies, which additionally shows that there is need for additional empirical studies, especially in an industrial context.

Group 2-Specific to the domain (testing embedded software)

- **RQ 2.1-Level of testing:** Most papers (202) consider system testing, which reflects the focus on testing the software of embedded systems as a whole. Furthermore, 32 papers focus on integration testing and 78 papers unit testing.
- **RQ 2.2-Types of test activity:** There is a good mix of papers proposing techniques and tools for each of the test activities. However, there is a focus on test execution (161 papers), criteria-based test-case design (161 papers), and test automation (142 papers). Also test evaluation (118 papers) and human-knowledge-based test-case design (68 papers) are well represented. Only a small number of papers address test management (20 papers) and other activities (12).
- **RQ 2.3-Types of test artifacts generated:** The largest ratio of papers (131) propose techniques to derive test-case inputs. 93 papers, address the generation of automated test code, 85 papers present approaches for generation of test case requirements, 73 papers discuss test oracles, and finally 9 papers proposed other types of generated test artifacts.
- **RQ 2.4-Type of non-functional testing:** Although the focus of this article was only to include functional testing papers, 25 papers additionally conducted performance testing, 17 real-time, 10 reliability testing, and 12 other types of non-functional testing including security testing.
- **RQ 2.5-Techniques to derive test artifacts:** The by-far most common technique is requirement-based testing (164 papers) including model-based testing, which is quite prominent in testing embedded software and covered by half of the papers. In addition, 47 papers were on white-box testing (code-coverage analysis), 21 papers on risk/fault-based testing, 21 papers on search-based testing, 23 papers on random testing, and 23 papers were about other techniques. As reported in Section 6.2.5, for a practitioner who is interested to adopt some of the presented techniques, it is important to know the advantages and disadvantages of using different testing techniques, whether these techniques are practical, and how much manual effort is involved in applying them. To objectively assess such important aspects, one needs to conduct empirical studies using those techniques on a set of the "same" SUTs and measure meaningful (and objective) metrics. However unfortunately, such critical and "comparative" analysis are quite rare, i.e., only one paper in the pool [Source 3] reported a comparative study of manual and automated testing in industrial control software. Thus, we would like to raise the need for further such studies in the future work.
- **RQ 2.6-Types of models used in model-based testing:** Model-based testing is quite common for testing embedded software. Test models are often specified as Finite State Machines (46 papers) or MATLAB Simulink models (28 papers). However, 75 papers use other types of test models including UML activity and sequence diagrams, which shows that there is a broad variety of model types used in model-based testing of embedded software.
- **RQ 2.7-Testing tools:** 95 papers used existing test tools and 85 papers proposed new test tools. Test tools therefore play a prominent role in papers on testing embedded software. This is in line with the result of RQ 1.1 that most contributions are test tools or frameworks.
- **RQ 2.8-Type of evaluation methods:** Examples showing the applicability of test approaches are provided in 149 papers, coverage of code or models in 66 papers, detecting real faults in 29 papers, detecting artificial faults in 50 papers, time/performance of test approaches in 53 papers, and other evaluation methods in 23 papers. One can see that evaluation via providing examples is dominating, which is a common evaluation method especially for formal methods. However, more studies providing stronger evaluation of effectiveness (detection of faults) and efficiency (time/performance) are needed.
- **RQ 2.9-Operating systems:** The most often reported operating systems are Unix/Linux (in 20 papers), vxWorks(in 5 papers), the Windows family (in 5 papers). Other operating systems are reported in overall 16 papers and include QNX, MicroC/OS-II, and Symbian OS.

Group 3-Specific to system under testing (SUT)

- **RQ 3.1-Simulated or real systems:** The usage of real systems is dominating (124 papers), but also simulated systems, i.e., Model-in-the-Loop (68 papers), Hardware-in-the-Loop (39 papers), Software-in-the-Loop (47 papers), and Processor-in-the-Loop (16 papers) are commonly used when testing embedded software.
- **RQ 3.2-Number of SUTs:** Most papers (164) applied the presented approach to one SUT only. But also two or three SUTs are quite common and applied in 26 and 17 papers, respectively.
- **RQ 3.3-Type/scale of SUTs:** Almost equal number of papers (115 and 114) experimented their proposed ideas with academic experimental and real commercial systems.
- **RQ 3.4-SUT programming/development languages:** Not surprisingly, C is the most popular programming language (70 papers). MATLAB/Simulink (36 papers), C++ (26 papers), and Java (11 papers) are popular as well. Examples for other programming languages are assembly language, C#, HDL, and Verilog.
- **RQ 3.5-SUT and board name:** 216 papers specified the SUT name explicitly, and 35 papers the hardware board name that they use.
- **RQ 3.6-Application domains/industries:** The automotive sector is the most active one in publishing research results in embedded software testing (81 papers). Industrial automation and control domain is the second (33 papers). Aviation, avionics and space are the third (28 papers). Other less common domains are mobile and telecom, home appliances and entertainment, medical, defense and railway.

Group 3-Trends and demographics:

- **RQ 4.1-Affiliation types of the study authors:** 151 papers were authored solely by academic researchers, 43 paper by practitioners only, and 78 papers as joint (collaborative) work between researchers and practitioners. Papers solely by academic authors often cover formal methods or other rather theoretical approaches like swarm optimization to test case prioritization. Collaborative work comprises many industrial case studies and frameworks. Finally, papers solely by industrial authors often cover practical issues, for instance on test process improvement or test-driven development. However, the transition of topics covered by academic, collaborative and industrial papers is smooth.
- **RQ 4.2-Active companies:** The most active companies in conducting research and publishing papers in testing embedded software are Daimler (14 papers), Samsung (8 papers), as well as Nokia and Berner & Mattner (4 papers each).
- **RQ 4.3-Citation analysis and highly-cited papers:** There are five papers on testing embedded software with more than 50 citations. Taking into account that this SLM already considers 312 papers, these citation numbers are not very high. The topics of the top-cited papers are quite diverse and there seem to be many isolated approaches but so far not an integrated view on testing embedded software. This SLM should be a cornerstone towards summarizing the advances in and synthesizing the entire field.

7.2 Benefits of this review

The authors have already started to benefit from the results of this review. In our ongoing collaborations with several industry partners in Turkey and Austria in the area of testing embedded software, our colleagues and we had various challenges in testing embedded software, e.g., in a project as published in [1], and we were quite uncertain of whether certain techniques already exist or whether we shall develop new techniques to solve the challenges. This summary and classification of the literature provided by review study was useful for us, by serving as an “index” to the vast body of knowledge in this important area. In this context, thanks to our review study, we are currently assessing several existing model-based techniques selected from the literature based on the review at hand for possible adoption/extension in our industry projects.

Also, as per our observations, unfortunately, many companies do not have an adequate overview of state-of-the–art and -practice in the area of testing embedded software. As a consequence, companies often reinvent the wheel in this area by designing a test approach new to them, but existing in the literature. This review paper and its supplementary material (the online repository and classification of all the 312 papers) close this gap and are intended to support the large world-wide community of software engineers and testers in the embedded software industry.

To further assess the benefits of this review, we asked two active test engineers in the Turkish embedded software industry to review this paper and the online spreadsheet of papers, and provide qualitative feedback to us on what they think about the potential benefits of the review. Only one of them had the time to study the review paper and the pool of studies and provide qualitative feedback to us. The following is the quote response from that test engineer: "*Our company conducts embedded system testing for software-intensive systems in the military domain. In such a context, one of our major problems is to borrow the actual Systems Under Test (SUT) from our customers because of security concerns. After reviewing this study, I wonder about "why we do not have any model of these SUTs". If we can create models of these SUTs (e.g., model-in-the loop, MIL), in the future we would not need to borrow the real systems from our customers and models may be sufficient for our test-case design and other test activities. There are a lot of studies in the pool of this review study, which would benefit us. I think this idea is a major benefit for companies like ours. And I hope we may realize this idea [in near future]. And as you have said in the paper, I believe that many companies are exactly reinventing wheel*". Note that we did not conduct a "formal" study to solicit the opinions of a large population of practitioners about this review paper, but such a study can be conducted in future.

7.3 Limitations and potential threats to validity

The main issues related to threats to validity of this SLM review are inaccuracy of data extraction, and incomplete set of studies in our pool due to limitation of search terms, selection of academic search engines, and researcher bias with regards to exclusion/inclusion criteria. In this section, these threats are discussed in the context of the four types of threats to validity based on a standard checklist for validity threats presented in [76]: internal validity, construct validity, conclusion validity and external validity. We discuss next those validity threats and the steps that we have taken to minimize or mitigate them.

Internal validity: The systematic approach that has been utilized for article selection is described in Section 4. In order to make sure that this review is repeatable, search engines, search terms and inclusion/exclusion criteria are carefully defined and reported. Problematic issues in selection process are limitation of search terms and search engines, and bias in applying exclusion/inclusion criteria.

Limitation of search terms and search engines can lead to incomplete set of primary sources. Different terms have been used by different authors to point to a similar concept. In order to mitigate risk of finding all relevant studies, formal searching using defined keywords has been done followed by manual search in references of initial pool and in web pages of active researchers in our field of study. For controlling threats due to search engines, not only we have included comprehensive academic databases such as Google Scholar. Therefore, we believe that adequate and inclusive basis has been collected for this study and if there is any missing publication, the rate will be negligible.

Applying inclusion/exclusion criteria can suffer from researchers' judgment and experience. Personal bias could be introduced during this process. To minimize this type of bias, joint voting is applied in article selection and only articles with high score are selected for this study.

Construct validity: Construct validities are concerned with issues that to what extent the object of study truly represents theory behind the study [76]. Threats related to this type of validity in this study were suitability of RQs and categorization scheme used for the data extraction. To limit construct threats in this study, GQM approach is used to preserve the tractability between research goal and questions.

Conclusion validity: Conclusion validity of a SLM study provided when correct conclusion reached through rigorous and repeatable treatment. In order to ensure reliability of our treatments, an acceptable size of primary sources is selected and terminology in defined schema reviewed by authors to avoid any ambiguity. All primary sources are reviewed by at least two authors to mitigate bias in data extraction. Each disagreement between authors was resolved by consensus among researchers. Following the systematic approach and described procedure ensured replicability of this study and assured that results of similar study will not have major deviations from our classification decisions.

External validity: External validity is concerned with to what extent the results of our systematic literature mapping can be generalized. As described in Section 4, we included scientific literature in the scope of testing embedded software with a sound validation written in English. The issue lies in whether our selected works can represent all types of literature in the area of testing embedded software. For this issue, we argue that relevant literature we selected in our pool taking scientific and grey literature into account contained sufficient information to represent the knowledge reported by previous researchers or professionals. As it can be seen from Section 6.3.1, the collected primary studies contain a significant proportion of academic, industrial and collaborative work which forms an adequate basis for concluding results useful for academia and applicable in industry. Also, note that our findings in this study are mainly within the field of testing embedded software. Beyond this field, we had no intention to generalize our results. Therefore, few problems with external validity are worthy of substantial attention.

8 CONCLUSIONS AND FUTURE WORK

To identify the state-of-the-art and the –practice in this area and to find out what we know about testing embedded software, we conducted and presented in this article a systematic literature mapping. Our article aims to benefit the readers (both practitioners and researchers) in providing the most comprehensive survey of the area of embedded software testing.

This paper investigates types of testing topics studied, types of testing activity, types of test artifacts generated, and the types of industries in which studies have focused, in embedded software testing based on a systematic literature review including 312 papers. Testing embedded software is a growing field of research where not only academia but also many companies especially from the automotive industry are active. Most studies for embedded software testing are available on test automation and especially on model-based testing. We recommend companies, who plan respective improvement measures, to take the collected body of knowledge from this SLM into account. Although there has been a high interest and progress in embedded software testing, more than half of the available papers only propose solution or report experiences, but do not provide empirical evidence on the effectiveness and efficiency of specific embedded software testing approaches. Therefore in future there is a need for more empirical studies providing industrial evidence on the effectiveness (e.g., measured in terms of defect detection rate) and efficiency of embedded software testing approaches in specific contexts to further improve decision support on the selection of embedded software testing approaches beyond this SLM.

Our future work includes using the findings of this SLM in our industry-academia collaborative projects, a more in-depth synthesis of the available evidence on testing embedded software and empirical evaluation of effectiveness and efficiency of available testing approaches for embedded software.

REFERENCES

The references section is divided into two parts: (1) Citations to the 312 papers reviewed in this review study; and (2) Other (regular) references cited throughout the paper.

8.1 SOURCES REVIEWED IN THE LITERATURE REVIEW

[Source 1] C. Baek, J. Jang, G. Jung, K. Choi, and S. Park, "A case study of black-box testing for embedded software using test automation tool," Journal of Computer Science, vol. 3, no. 3, pp. 144-148, 2007.

[Source 2] T. Sekizawa and T. Kotorii, "A case study: Verification of specifications of an embedded system and generation of verification items using pairwise testing," in Software Engineering Conference, 2013, pp. 146-151.

[Source 3] E. P. Enoiu, D. Sundmark, A. Causevic, and P. Pettersson, "A Comparative Study of Manual and Automated Testing in Industrial Embedded Software," in International Conference on Testing Software and Systems, , pp. 412-417, Graz, Sweden, 2016.

[Source 4] E. P. Enoiu, A. Cauevic, D. Sundmark, and P. Pettersson, "A Controlled Experiment in Testing of Safety-Critical Embedded Software," IEEE International Conference on Software Testing, Verification and Validation, 2016, pp. 1-11.

[Source 5] P. M. Kruse, J. Wegener, and S. Wappler, "A cross-platform test system for evolutionary black-box testing of embedded systems," ACM SIGEVOlution, vol. 5, pp. 3-9, 2010.

[Source 6] B. Kang, Y.-J. Kwon, and R. Y. Lee, "A design and test technique for embedded software," International Conference on Software Engineering Research, Management and Applications, 2005, pp. 160-165.

[Source 7] G. Wang, F. Miao, W. Zhang, and H. Yu, "A dynamic and interactive diagnosing and testing method for development of digital TV receiver system," in International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, , 2007, pp. 1107-1112.

[Source 8] H.-m. Qian and C. Zheng, "An embedded software testing process model," International Conference on Computational Intelligence and Software Engineering, 2009, pp. 1-5.

[Source 9] P. Runeson, P. Heed, and A. Westrup, "A Factorial Experimental Evaluation of Automated Test Input Generation," Product-Focused Software Process Improvement, pp. 217-231, 2011.

[Source 10] A. Guerrouat and H. Richter, "A formal approach for analysis and testing of reliable embedded systems," Electronic Notes in Theoretical Computer Science, vol. 141, pp. 91-106, 2005.

[Source 11] P. M. Kruse, J. Wegener, and S. Wappler, "A highly configurable test system for evolutionary black-box testing of embedded systems," in Proceedings of Annual conference on Genetic and evolutionary computation, 2009, pp. 1545-1552.

[Source 12] W. Chaaban, M. Schwarz, B. Batchuluun, H. Sheng, and J. Börcsök, "A HiL test bench for verification and validation purposes of model-based developed applications using Simulink® and OPC DA technology," International Conference on Electrical and Electronics Engineering, 2011, pp. II-56-II-61.

[Source 13] R. M. Macieira and E. Barros, "A Mechanism for Monitoring Driver-Device Communication," in Embedded Software Verification and Debugging, Springer, 2017, pp. 133-158.

[Source 14] A.-R. Guduvan, H. Waeselynck, V. Wiels, G. Durrieu, Y. Fusero, and M. Schieber, "A Meta-model for Tests of Avionics Embedded Systems," International Conference on Model-Driven Engineering and Software Development, 2013, pp. 5-13.

[Source 15] D. Hura and M. Dimmich, "A method facilitating integration testing of embedded software," in Proceedings of International Workshop on Dynamic Analysis, 2011, pp. 7-11.

[Source 16] Y. Yin and B. Liu, "A method of test case automatic generation for embedded software," International Conference on Information Engineering and Computer Science, 2009, pp. 1-5.

[Source 17] J. Ye, W. Dong, and Z. Qi, "A method to generate embedded real-time system test suites based on software architecture specifications," International Conference for Young Computer Scientists, 2008, pp. 2325-2329.

[Source 18] Y.-w. Dong, G. Wang, and H.-b. Zhao, "A Model-based Testing for AADL Model of Embedded Software," International Conference on Quality Software, 2009, pp. 185-190.

[Source 19] R. Marinescu, M. Saadatmand, A. Bucaioni, C. Seceleanu, and P. Pettersson, "A model-based testing framework for automotive embedded systems," EUROMICRO Conference on Software Engineering and Advanced Applications, 2014, pp. 38-47.

[Source 20] C.-H. Liu, S.-L. Chen, and T.-C. Huang, "A Model-Based Testing Tool for Embedded Software," International Conference on Genetic and Evolutionary Computing, 2012, pp. 180-183.

[Source 21] M. Broy and A. Pretschner, "A Model-Based View onto Testing: Criteria for the Derivation of Entry Tests for Integration Testing," Model-Based Testing for Embedded Systems, CRC Press, pp. 245, 2011.

[Source 22] H. Lei and Y. Wang, "A model-driven testing framework based on requirement for embedded software," International Conference on Reliability, Maintainability and Safety, 2016, pp. 1-6.

[Source 23] Y. Ren and T. Li, "A new method for testing embedded software with scenario pattern," International Conference on Technologies for E-Learning and Digital Entertainment, 2006, pp. 178-182.

[Source 24] Q. Shen, Y. Jiang, and J. Lou, "A new test suite reduction method for wearable embedded software," Computers & Electrical Engineering, vol. 61, pp. 116-125, 2017.

[Source 25] Y. Cho and C.-W. Yoo, "A performance profile and test tool for development of embedded software using various report views," Computational Science, pp. 510-517, 2006.

[Source 26] R. Awedikian and B. Yannou, "A practical model-based statistical approach for generating functional test cases: application in the automotive industry," Software Testing, Verification and Reliability, vol. 24, pp. 85-123, 2014.

[Source 27] J. Seo, B. Choi, and S. Yang, "A profiling method by PCB hooking and its application for memory fault detection in embedded system operational test," Information and Software Technology, vol. 53, pp. 106-119, 2011.

[Source 28] S. Biswas, R. Mall, and M. Satpathy, "A regression test selection technique for embedded software," ACM Transactions on Embedded Computing Systems), vol. 13, p. 47, 2013.

[Source 29] F. Lindlar and A. Windisch, "A search-based approach to functional hardware-in-the-loop testing," International Symposium on Search-Based Software Engineering, 2010, pp. 111-119.

[Source 30] A. Venticinque, N. Mazzocca, and S. Venticinque, "A semantic support for testing activities of safety-critical embedded systems," International Conference on Complex, Intelligent and Software Intensive Systems, 2014, pp. 576-581.

[Source 31] O. Ballan, P. Bernardi, B. Yazdani, and E. Sánchez, "A software-based self-test strategy for on-line testing of the scan chain circuitries in embedded microprocessors," IEEE International on On-Line Testing Symposium, 2013, pp. 79-84.

[Source 32] V. Legourski, C. Trödhandl, and B. Weiss, "A system for automatic testing of embedded software in undergraduate study exercises," ACM SIGBED (Special Interest Group on Embedded Systems) Review, vol. 2, pp. 48-55, 2005.

[Source 33] M. Conrad, "A systematic approach to testing automotive control software," Convergence International Congress & Exposition On Transportation Electronics, 2004.

[Source 34] A. Krupp and W. Müller, "A systematic approach to the test of combined HW/SW systems," in Proceedings of the Conference on Design, Automation and Test in Europe, 2010, pp. 323-326.

[Source 35] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing," 2006.

[Source 36] E. Jung, "A test process improvement model for embedded software developments," in Quality Software, 2009. QSIC'09. 9th International Conference on, 2009, pp. 432-437.

[Source 37] M. Smith, J. Miller, and S. Daeninck, "A test-oriented embedded system production methodology," Journal of Signal Processing Systems , vol. 56, pp. 69-89, 2009.

[Source 38] D. Kwack, Y. Cho, J. Choi, and C.-W. Yoo, "A XML-based Testing Tool for Embedded Software," in Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on, 2007, pp. 431-438.

[Source 39] A. Guerrouat and H. Richter, "Adaptation of State/Transition-Based Methods for Embedded System Testing," World Academy of Science, Engineering and Technology, vol. 10, 2005.

[Source 40] C. Lee, "Adapting and adjusting test process reflecting characteristics of embedded software and industrial properties based on referential models," in Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, 2009, pp. 1372-1377.

[Source 41] W.-T. Tsai, Y. Na, R. Paul, F. Lu, and A. Saimi, "Adaptive scenario-based object-oriented test frameworks for testing embedded systems," in Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International, 2002, pp. 321-326.

[Source 42] R. D. C. C. Soldi and A. A. M. Frohlich, "AEP-Automatic Exchange of Embedded System Software Parameters," in High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on, 2013, pp. 1938-1943.

[Source 43] W. Chongwen, D. Gangyi, M. Rui, and Z. Hongzhe, "All Digital Embedded Software Simulation Testing Environment based on component," in Computer Design and Applications (ICCDA), 2010 International Conference on, 2010, pp. V5-449-V5-453.

[Source 44] G. Din, K.-D. Engel, and A. Rennoch, "An Approach for test derivation from system architecture models applied to embedded systems," Model-based Testing in Practice, pp. 23-32, 2009.

[Source 45] T. Yu, A. Sung, W. Srisa-An, and G. Rothermel, "An approach to testing commercial embedded systems," Journal of Systems and Software, vol. 88, pp. 207-230, 2014.

[Source 46] P. Iyenghar, C. Westerkamp, J. Wuebbelmann, and E. Pulvermueller, "An architecture for deploying model based testing in embedded systems," Forum on Specification & Design Languages , 2010, pp. 1-6.

[Source 47] H. S. Chae, G. Woo, T. Y. Kim, J. H. Bae, and W.-Y. Kim, "An automated approach to reducing test suites for testing retargeted C compilers for embedded systems," Journal of Systems and Software, vol. 84, pp. 2053-2064, 2011.

[Source 48] B. Chetali and Q.-H. Nguyen, "An automated testing experiment for layered embedded C code," International Journal on Software Tools for Technology Transfer , vol. 11, pp. 175-185, 2009.

[Source 49] Y. Shuaishuai, Y. Zhengwei, L. Bin, L. Yunfeng, and G. Zhijie, "An automatic testing framework for embedded software," International Conference on Computer Science and Education, 2017, pp. 269-274.

[Source 50] J. J. Li, D. M. Weiss, and H. Yee, "An automatically-generated run-time instrumenter to reduce coverage testing overhead," in Proceedings international workshop on Automation of software test, 2008, pp. 49-56.

[Source 51] A. Chunduri, "An Effective Verification Strategy for Testing Distributed Automotive Embedded Software Functions", International Conference on Product-Focused Software Process Improvement, pp. 233-248, 2016

[Source 52] Y. Cho and J. Choi, "An embedded software testing tool supporting multi-paradigm views," International Conference on Computational Science and Its Applications, 2008, pp. 780-789.

[Source 53] V. de Oliveira Neves, M. E. Delamaro, and P. C. Masiero, "An environment to support structural testing of autonomous vehicles," Brazilian Symposium on Computing Systems Engineering, 2014, pp. 19-24.

[Source 54] S. Weißleder and H. Schlingloff, "An evaluation of model-based testing in embedded applications," IEEE International Conference on Software Testing, Verification and Validation, 2014, pp. 223-232.

[Source 55] J. Guan, J. Offutt, and P. Ammann, "An industrial case study of structural testing applied to safety-critical embedded software," Proceedings of ACM/IEEE international symposium on Empirical software engineering, 2006, pp. 272-277.

[Source 56] B. Marculescu, R. Feldt, R. Torkar, and S. Poulding, "An initial industrial evaluation of interactive search-based testing for embedded software," Applied Soft Computing, vol. 29, pp. 26-39, 2015.

[Source 57] A. Sung and B. Choi, "An interaction testing technique between hardware and software in embedded systems," Asia-Pacific Software Engineering Conference, 2002, pp. 457-464.

[Source 58] A. Sung, B. Choi, and S. Shin, "An interface test model for hardware-dependent software and embedded OS API of the embedded system," Computer Standards & Interfaces, vol. 29, pp. 430-443, 2007.

[Source 59] A. A. Khwaja, "An MFC based multi-threaded test environment for the validation of an embedded automotive microcontroller," Proceedings International Conference on Technology of Object-Oriented Languages and Systems, 2000, pp. 15-24.

[Source 60] F. Abbors, V.-M. Aho, K. Jani, R. Teittinen, and D. Truscan", Applying Model-Based Testing in the Telecommunication Domain," in Model-Based Testing for Embedded Systems, CRC Press, 2011, pp. 486-524.

[Source 61] K. H. S. Hla, Y. Choi, and J. S. Park, "Applying particle swarm optimization to prioritizing test cases for embedded real time software retesting," IEEE International Conference on Computer and Information Technology Workshops , 2008, pp. 527-532.

[Source 62] L. Lazić and D. Velašević, "Applying simulation and design of experiments to the embedded software testing process," Software Testing, Verification and Reliability, vol. 14, pp. 257-282, 2004.

[Source 63] J. Grenning, "Applying test-driven development to embedded software," IEEE Instrumentation & Measurement Magazine, vol. 10, no. 6, pp. 20-25, 2007.

[Source 64] H. Zipori, Z. Sagiv, and A. Yossovitch, "Approaches and implementation of software test and development system for embedded computer systems," Proceedings Israel Conference on Computer Systems and Software Engineering, 1988, pp. 30-39.

[Source 65] A. Costin, A. Zarras, and A. Francillon, "Automated dynamic firmware analysis at scale: a case study on embedded web interfaces," in Proceedings of ACM Asia Conference on Computer and Communications Security, 2016, pp. 437-448.

[Source 66] C. P. Vudatha, S. Nalliboena, S. K. Jammalamadaka, B. K. K. Duvvuri, and L. Reddy, "Automated generation of test cases from output domain and critical regions of embedded systems using genetic algorithms," National Conference on Emerging Trends and Applications in Computer Science, 2011, pp. 1-6.

[Source 67] B. Cukic, B. J. Taylor, and H. Singh, "Automated generation of test trajectories for embedded flight control systems," International Journal of Software Engineering and Knowledge Engineering, vol. 12, pp. 175-200, 2002.

[Source 68] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, and C. Poull, "Automated model-in-the-loop testing of continuous controllers using search," in International Symposium on Search Based Software Engineering, 2013, pp. 141-157.

[Source 69] J. Hawkins, R. B. Howard, and H. V. Nguyen, "Automated real-time testing (ARTT) for embedded control systems (ECS)," Proceedings of Reliability and Maintainability Symposium, 2002, pp. 647-652.

[Source 70] C. Wang, "Automated requirements-driven testing of embedded systems based on use case specifications and timed automata," University of Luxembourg, Luxembourg, PhD thesis, 2017

[Source 71] J. Andersson and K. Andersson, "Automated Software Testing in an Embedded Real-Time System," MSc thesis, Linköping University, 2007.

[Source 72] J. H. Poore, L. Lin, R. Eschbach, and T. Bauer, "Automated statistical testing for embedded systems," Model-Based Testing for Embedded Systems in the Series on Computational Analysis and Synthesis, and Design of Dynamic Systems. CRC Press-Taylor & Francis, 2011

[Source 73] Muhammad Zohaib Iqbal, Andrea Arcuri, L. Briand, "Automated System Testing of Real-Time Embedded Systems Based on Environment Models", Simula Research Laboratory Technical Report (2011-19), 2011

[Source 74] C. Zhang, X. Bai, J. Li, and R. Zhang, "Automated test case generation for embedded software using extended interface automata," International Conference on Quality Software , 2013, pp. 292-298.

[Source 75] D.-H. Kum, J. Son, S.-b. Lee, and I. Wilson, "Automated testing for automotive embedded systems," in Conference of the Society of Instrument and Control Engineers of Japan (SICE-ICASE), 2006., 2006, pp. 4414-4418.

[Source 76] S. Siegl, K.-S. Hielscher, R. German, and C. Berger, "Automated testing of embedded automotive systems from requirement specification models," Latin American Test Workshop, 2011, pp. 1-6.

[Source 77] R. Ramler, W. Putschögl, and D. Winkler, "Automated testing of industrial automation software: practical receipts and lessons learned," in Proceedings of International Workshop on Modern Software Engineering Methods for Industrial Automation, 2014, pp. 7-16.

[Source 78] Y. Kim, Y. Kim, T. Kim, G. Lee, Y. Jang, and M. Kim, "Automated unit testing of large industrial embedded software using concolic testing," in Proceedings ofIEEE/ACM International Conference on Automated Software Engineering, 2013, pp. 519-528.

[Source 79] M. A. Wehrmeister and G. R. Berkenbrock, "Automatic Execution of Test Cases on UML Models of Embedded Systems," in International Embedded Systems Symposium, 2013, pp. 39-48.

[Source 80] C. Wang, F. Pastore, A. Goknil, L. Briand, and Z. Iqbal, "Automatic generation of system test cases from use case specifications," in Proceedings of International Symposium on Software Testing and Analysis, 2015, pp. 385-396.

[Source 81] S. Weißleder and H. Schlingloff, "Automatic model-based test generation from UML state machines," Model-Based Testing for Embedded Systems, CRC Press, p. 77-110, 2011.

[Source 82] C. Murphy, Z. Zoomkawalla, and K. Narita, "Automatic test case generation and test suite reduction for closed-loop controller software," University of Pennsylvania, Technical Report SD-008, 2013.

[Source 83] P. Mani and M. Prasanna, "Automatic test case generation for programmable logic controller using function block diagram," International Conference on Information Communication and Embedded Systems2016, pp. 1-4.

[Source 84] P. Dämon, "Automatic Test Case Generation for Safety-Related Embedded Systems," SAE International Journal of Passenger Cars-Electronic and Electrical Systems, vol. 1, pp. 18-25, 2008.

[Source 85] P. Bokil, P. Darke, U. Shrotri, and R. Venkatesh, "Automatic test data generation for c programs," IEEE International Conference on Secure Software Integration and Reliability Improvement,, 2009, pp. 359-368.

[Source 86] J. Wegener, K. Buhr, and H. Pohlheim, "Automatic test data generation for structural testing of embedded software systems by evolutionary testing," in Proceedings of Annual Conference on Genetic and Evolutionary Computation, 2002, pp. 1233-1240.

[Source 87] Y. Zhang, H. Li, and X. Li, "Automatic test program generation using executing-trace-based constraint extraction for embedded processors," IEEE Transactions on Very Large Scale Integration Systems, vol. 21, pp. 1220-1233, 2013.

[Source 88] C.-S. Koong, C. Shih, P.-A. Hsiung, H.-J. Lai, C.-H. Chang, W. C. Chu, et al., "Automatic testing environment for multi-core embedded software—ATEMES," Journal of Systems and Software, vol. 85, pp. 43-60, 2012.

[Source 89] T. Nakajima, Y. Bessho, H. Yamanaka, and K. Hirota, "Automatic testing of embedded software based on state-transition requirement specifications," Electronics and Communications in Japan (Part II: Electronics), vol. 86, pp. 64-75, 2003.

[Source 90] V. Papailiopoulou, B. Seljimi, and I. Parissis, "Automatic testing of Lustre/SCADE programs," Model-Based Testing for Embedded Systems, CRC Press, pp. 171-194, 2011.

[Source 91] J. Seo, A. Sung, B. Choi, and S. Kang, "Automating embedded software testing on an emulated target board," in Proceedings of International Workshop on Automation of Software Test, 2007, p. 9.

[Source 92] S. J. Cunning and J. W. Rozenblit, "Automating test generation for discrete event oriented embedded systems," Journal of Intelligent & Robotic Systems, vol. 41, pp. 87-112, 2005.

[Source 93] H. Moon, G. Kim, Y. Kim, S. Shin, K. Kim, and S. Im, "Automation test method for automotive embedded software based on AUTOSAR," International Conference on, Software Engineering Advances2009, pp. 158-162.

[Source 94] A. Arcuri, M. Z. Z. Iqbal, and L. C. Briand, "Black-Box System Testing of Real-Time Embedded Systems Using Random and Search-Based Testing," International Conference on Testing Software and Systems , vol., pp. 95-110, 2010.

[Source 95] M. Bell, G. Danialy, M. Howells, and S. Pateras, "Bridging the gap between embedded test and ATE," Proceedings of International Test Conference, 2000, pp. 55-63.

[Source 96] M. Burford and F. Belli, "CADAS: A Tool for Rapid Prototyping and Testing of Embedded Software," in IEEE Proceedings on Database Conference, 1983, pp. 23-26.

[Source 97] M. Wahler, E. Ferranti, R. Steiger, R. Jain, and K. Nagy, "CAST: Automating software tests for embedded systems," IEEE International Conference on Software Testing, Verification and Validation , 2012, pp. 457-466.

[Source 98] R. Matinnejad, S. Nejati, L. Briand, and T. Brcukmann, "CoCoTest: a tool for model-in-the-loop testing of continuous controllers," in Proceedings of ACM/IEEE international conference on Automated software engineering, 2014, pp. 855-858.

[Source 99] L. Xiao, M. Li, M. Gu, and J. Sun, "Combinational testing of PLC programs based on the denotational semantics of instructions," World Congress on Intelligent Control and Automation, 2014, pp. 5840-5845.

[Source 100] G. Dhadyalla, N. Kumari, and T. Snell, "Combinatorial testing for an automotive hybrid electric vehicle control system: a case study," IEEE International Conference on Software Testing, Verification and Validation, 2014, pp. 51-57.

[Source 101] M. Iqbal, A. Arcuri, and L. Briand, "Combining search-based and adaptive random testing strategies for environment model-based testing of real-time embedded systems," International Symposium on Search Based Software Engineering, pp. 136-151, 2012.

[Source 102] V. Yaneva, A. Rajan, and C. Dubach, "Compiler-assisted test acceleration on gpus for embedded software," in Proceedings of ACM International Symposium on Software Testing and Analysis, 2017, pp. 35-45.

[Source 103] M. Kooli, F. Kaddachi, G. Di Natale, A. Bosio, P. Benoit, and L. Torres, "Computing reliability: On the differences between software testing and software fault injection techniques," Microprocessors and Microsystems, vol. 50, pp. 102-112, 2017.

[Source 104] T. Kim, J. Park, I. Kulida, and Y. Jang, "Concolic Testing Framework for Industrial Embedded Software," Asia-Pacific Software Engineering Conference, 2014, pp. 7-10.

[Source 105] Y. Kim, M. Kim, and Y. Jang, "Concolic testing on embedded software-case studies on mobile platform programs," in European Software Engineering Conference/Foundations of Software Engineering , 2011, p. 30.

[Source 106] N. H. Lee, T. H. Kim, and S. D. Cha, "Construction of global finite state machine for testing task interactions written in message sequence charts," in Proceedings of international conference on Software engineering and knowledge engineering, 2002, pp. 369-376.

[Source 107] I. Schieferdecker, E. Bringmann, and J. Großmann, "Continuous TTCN-3: testing of embedded control systems," in Proceedings of international workshop on Software engineering for automotive systems, 2006, pp. 29-36.

[Source 108] R. Mitsching, C. Weise, D. Franke, T. Gerlitz, and S. Kowalewski, "Coping with Complexity of Testing Models for Real-Time Embedded Systems," International Conference on Secure Software Integration & Reliability Improvement Companion, 2011, pp. 128-135.

[Source 109] X. Wu, J. J. Li, D. Weiss, and Y. Lee, "Coverage-based testing on embedded systems," International Workshop on Automation of Software Test, 2007, pp. 7-7.

[Source 110] J. C. Costa and J. C. Monteiro, "Coverage-directed observability-based validation for embedded software," ACM Transactions on Design Automation of Electronic Systems, , vol. 18, p. 19, 2013.

[Source 111] Z. Havlice, V. Szabóová, and J. Vízi, "Critical knowledge representation for model-based testing of embedded systems," IEEE International Symposium on Applied Machine Intelligence and Informatics, 2013, pp. 169-174.

[Source 112] M. Broy, S. Chakraborty, D. Goswami, S. Ramesh, M. Satpathy, S. Resmerita, et al., "Cross-layer analysis, testing and verification of automotive control software," in Proceedings of ACM international conference on Embedded software, 2011, pp. 263-272.

[Source 113] C. A. Conceicao, F. Mattiello-Francisco, and C. L. Batista, "Dependability verification of nanosatellite embedded software supported by a reusable Test System," Latin-American Symposium on Dependable Computing, 2016, pp. 157-163.

[Source 114] P. Caliebe, T. Herpel, and R. German, "Dependency-based test case selection and prioritization in embedded systems," IEEE International Conference on Software Testing, Verification and Validation, 2012, pp. 731-735.

[Source 115] J. Zander-Nowicka, I. Schieferdecker, and T. Farkas, "Derivation of Executable Test Models from Embedded System Models using Model Driven Architecture Artefacts-Automotive Domain," in Workshop on Model-Based Development of Embedded Systems (MBEES), 2006, pp. 131-140.

[Source 116] W. Zhong-Min, H. Yi-Wei, and L. Chen, "Design and implementation of a static test approach for embedded software," World Congress on Software Engineering, 2012, pp. 125-127.

[Source 117] W.-j. Han, H. Yan, and Z.-h. Dong, "Design and implementation of the embedded software testing platform for the gun fire control system," International Conference on Computational Intelligence and Software Engineering,, 2009, pp. 1-4.

[Source 118] S. P. Karmore and A. R. Mahajan, "Development of Novel Approach for Testing of Differential Embedded Systems," International Conference on Emerging Trends in Engineering and Technology, 2013, pp. 46-47.

[Source 119] S. P. Karmore, A. R. Mahajan, and G. C. Jarbias, "Development of software interface for testing of embedded system," International Conference on Advanced Computing Technologies, 2013, pp. 1-6.

[Source 120] P. Daniel and R. Chandel, "Dynamic Ruleset Based Online Concurrent Testing of Functional Faults for Embedded Controllers," Przegląd Elektrotechniczny, vol. 90, pp. 111--114, 2014.

[Source 121] N. Bartzoudis, V. Tantsios, and K. McDonald-Maier, "Dynamic scheduling of test routines for efficient online self-testing of embedded microprocessors," IEEE International On-Line Testing Symposium, 2008, pp. 185-187.

[Source 122] L. Pierre and L. Ferro, "Dynamic Verification of SystemC Transactional Models," CRC press, 2011.

[Source 123] M. Smith, A. Kwan, A. Martin, and J. Miller, "E TDD Ō Embedded Test Driven Development a Tool for Hardware Software Co design Projects," Extreme Programming and Agile Processes in Software Engineering, pp. 145, 2005.

[Source 124] M. J. Karlesky, W. I. Bereza, and C. B. Erickson, "Effective test driven development for embedded software," in IEEE International Conference on Electro/information Technology, 2006, pp. 382-387.

[Source 125] P. Krishnan, R. Venkatesh, P. Bokil, T. Muske, and V. Suman, "Effectiveness of random testing of embedded systems," in International Conference on System Science, 2012, pp. 5556-5563.

[Source 126] D. Sundmark, A. Pettersson, S. Eldh, M. Ekman, H. Thane, and A. Ericsson, "Efficient system-level testing of embedded real-time software," Euromicro Conference on Real-Time Systems, 2007, p. 53-56.

[Source 127] T. L. Bennett and P. W. Wennberg, "Eliminating embedded software defects prior to integration test," Crosstalk, Journal of Defence Software Engineering, 2005.

[Source 128] R. Figueiredo, P. A. Dinda, and J. Fortes, "Guest editors' introduction: Resource virtualization renaissance," IEEE Computer, vol. 38, pp. 28-31, 2005.

[Source 129] R. P. Pontes, E. Martins, A. M. Ambrósio, and E. Villani, "Embedded critical software testing for aerospace applications based on PUS," in Proceedings of Brazilian Symposium on Computer Networks and Distributed Systems, Brazil, 2010.

[Source 130] K. R. Leung, J.-Y. Ng, and W. Yeung, "Embedded program testing in untestable mobile environment: an experience of trustworthiness approach," Asia-Pacific Software Engineering Conference, 2004, pp. 430-437.

[Source 131] Y. Wu, S. Wang, and Z. Yu, "Embedded software reliability testing and its practice," in International Conference on Computer Design and Applications, pp. V2-24-V2-27.

[Source 132] P. R. Maier and V. B. Kleeberger, "Embedded software reliability testing by unit-level fault injection," in Asia and South Pacific Design Automation Conference, 2016, pp. 410-416.

[Source 133] A. Dube, "Embedded Software Testability: How to design for it and How to measure the degree to which it has been achieved", MSc thesis, University of Amsterdam, 2013.

[Source 134] J. Lepistö, "Embedded Software Testing Methods", BSc thesis, Helsinki Metropolia University of Applied Sciences, 2012.

[Source 135] T. Arons, E. Elster, T. Murphy, and E. Singerman, "Embedded software validation: Applying formal techniques for coverage and test generation," International Workshop on Microprocessor Test and Verification, 2006, pp. 45-51.

[Source 136] M.-H. Lee, C.-J. Yoo, and O.-B. Jang, "Embedded system software testing based on SOA for mobile service," International Journal of Advanced Science and Technology, vol. 1, pp. 55-64, 2008.

[Source 137] E. Bergersen, "Embedded Unit Testing Framework," ed: MSc thesis, Norwegian University of Science and Technology, 2013.

[Source 138] M. Z. Iqbal, A. Arcuri, and L. Briand, "Environment modeling and simulation for automated testing of soft real-time embedded software," Software & Systems Modeling, vol. 14, pp. 483-524, 2015.

[Source 139] S. Liu, "Evaluation of Model-Based Testing for Embedded Systems based on the Example of the Safety-Critical Vehicle Functions", MSc thesis, University of Gothenburg, 2012.

[Source 140] T. E. Vos, F. F. Lindlar, B. Wilmes, A. Windisch, A. I. Baars, P. M. Kruse, et al., "Evolutionary functional black-box testing in an industrial setting," Software Quality Journal, vol. 21, pp. 259-288, 2013.

[Source 141] H. Pohlheim, M. Conrad, and A. Griep, "Evolutionary safety testing of embedded control software by automatically generating compact test data sequences," Society of Automotive Engineers (SAE) Technical Paper 0148-7191, 2005.

[Source 142] R. Seinauskas and V. Seinauskas, "Examination of the possibilities for integrated testing of embedded systems," American journal of embedded systems and applications, vol. 1, pp. 1-12, 2013.

[Source 143] A. Ulrich and A. Votintseva, "Experience report: Formal verification and testing in the development of embedded software," IEEE International Symposium on Software Reliability Engineering, 2015, pp. 293-302.

[Source 144] A. Hoffmann, J. Quante, and M. Woehrle, "Experience report: White box test case generation for automotive embedded software," IEEE International Conference on Software Testing, Verification and Validation, 2016, pp. 269-274.

[Source 145] X. Yun, Y. Xiaoqiang, D. Zhuangzhi, and L. Ning, "Fault test device of electrical system based on embedded equipment," Journal of Theoretical & Applied Information Technology, vol. 45, 2012.

[Source 146] M. Esser and P. Struss, "Fault-Model-Based Test Generation for Embedded Software," International Joint Conference on Artificial Intelligence (I, 2007, pp. 342-347.

[Source 147] M. Lochau and U. Goltz, "Feature interaction aware test case generation for embedded control systems," Electronic Notes in Theoretical Computer Science, vol. 264, pp. 37-52, 2010.

[Source 148] S. Siegl, K.-S. Hielscher, R. German, and C. Berger, "Formal specification and systematic model-driven testing of embedded automotive systems," in Design, Automation & Test in Europe Conference & Exhibition, 2011, pp. 1-6.

[Source 149] Z. Li, B. Liu, N. Ma, and Y. Yin, "Formal testing applied in embedded software," International Conference on Reliability, Maintainability and Safety, 2009, pp. 697-702.

[Source 150] J. Prelgauskas and E. Bareisa, "Generating Unit Tests for Floating Point Embedded Software using Compositional Dynamic Symbolic Execution," Elektronika ir Elektrotechnika, vol. 122, pp. 19-22, 2012.

[Source 151] W. Mueller, A. Bol, A. Krupp, and O. Lundkvist, "Generation of executable testbenches from natural language requirement specifications for embedded real-time systems," Distributed, Parallel and Biologically Inspired Systems, pp. 78-89, 2010.

[Source 152] T. Reinbacher, A. Steininger, T. Müller, J. Brauer, and S. Kowalewski, "Hardware support for efficient testing of embedded software," International Conference on Mechatronic and Embedded Systems and Applications, pp. 1-10, 2011.

[Source 153] A. Motghare and S. P. Karmore, "Hardware-in-the-Loop Search–Based Testing Approach to Embedded Systems" International Journal of Recent Technology and Engineering, 2012

[Source 154] M. Utting, "How to design extended finite state machine test models in Java," Model-Based Testing for Embedded Systems, CRC Press, pp. 147-169, 2011.

[Source 155] Y.-D. Lin, E. T.-H. Chu, S.-C. Yu, and Y.-C. Lai, "Improving the accuracy of automated GUI testing for embedded systems," IEEE Software, vol. 31, pp. 39-45, 2014.

[Source 156] M. Kim, Y. Kim, and Y. Jang, "Industrial application of concolic testing on embedded software: Case studies," IEEE International Conference on Software Testing, Verification and Validation, 2012, pp. 390-399.

[Source 157] P. Iyenghar, E. Pulvermueller, C. Westerkamp, and J. Wuebbelmann, "Infrastructure support to convey test data from state diagrams for executing MBT in embedded systems," IEEE International Conference on Computer as a Tool", 2013, pp. 651-659.

[Source 158] A. M. Perez and S. Kaiser, "Integrating test levels for embedded systems," in Workshop on Testing: Academic and Industrial Conference-Practice and Research Techniques., 2009, pp. 184-193.

[Source 159] A. Sung and B. Choi, "Interaction testing in an embedded system using hardware fault injection and program mutation," in International Workshop on Formal Approaches to Software Testing, 2003, pp. 192-204.

[Source 160] B. Marculescu, "Interactive Search-Based Testing of Embedded Software: Understanding the Benefits and Difficulties of Industrial Application," BSc thesis, Blekinge Institute of Technology, 2014.

[Source 161] T. Bennett and P. Wennberg, "Maintaining verification test consistency between executable specifications and embedded software in a virtual system integration laboratory environment," Proceedings of Annual NASA Goddard Software Engineering Workshop, 2003, pp. 221-228.

[Source 162] X. Tai, "MFQ & PPDCS-Test Analysis and Test Design for Large Embedded Software Systems," International Conference on Software Engineering Advances, 2009, pp. 108-117.

[Source 163] M. Spieker, A. Noyer, P. Iyenghar, G. Bikker, J. Wuebbelmann, and C. Westerkamp, "Model based debugging and testing of embedded systems without affecting the runtime behaviour," IEEE Conference on Emerging Technologies & Factory Automation, 2012, pp. 1-6.

[Source 164] V. Chimisliu and F. Wotawa, "Model based test case generation for distributed embedded systems," IEEE International Conference on Industrial Technology , 2012, pp. 656-661.

[Source 165] M. Cavallaro and A. Marino, "Model Based Testing Applied to In-house Embedded Software Development," SAE Technical Paper 0148-7191, 2010.

[Source 166] T. Dang, "Model-based testing of hybrid systems," Model-Based Testing for Embedded Systems, CRC Press, pp. 383-424, 2010.

[Source 167] S. Siegl, K.-S. Hielscher, and R. German, "Model driven testing of embedded automotive systems with timed usage models," IEEE International Conference on Vehicular Electronics and Safety, 2010, pp. 110-115.

[Source 168] P. Iyenghar, E. Pulvermueller, C. Westerkamp, J. Wuebbelmann, and M. Uelschen, "Model-based debugging of embedded software systems," in Embedded Software Verification and Debugging,Springer, 2017, pp. 107-132.

[Source 169] F. Belli, A. Hollmann, and S. Padberg, "Model-Based Integration Testing with Communication Sequence Graphs," Model-Based Testing for Embedded Systems, CRC Press, pp. 223, 2011.

[Source 170] P. Skruch and G. Buchala, "Model-based real-time testing of embedded automotive systems," SAE International Journal of Passenger Cars-Electronic and Electrical Systems, vol. 7, pp. 337-344, 2014.

[Source 171] S. Yang, B. Liu, S. Wang, and M. Lu, "Model-based robustness testing for avionics-embedded software," Chinese Journal of Aeronautics, vol. 26, pp. 730-740, 2013.

[Source 172] Y. Wang and Y. Wang, "Model-Based Simulation Testing for Embedded Software," in Communications and Mobile Computing (CMC), 2011 Third International Conference on, 2011, pp. 103-109.

[Source 173] P. Iyenghar, J. Wuebbelmann, C. Westerkamp, and E. Pulvermueller, "Model-based test case generation by reusing models from runtime monitoring of deeply embedded systems," IEEE Embedded Systems Letters , vol. 5, pp. 38-41, 2013.

[Source 174] A. Mjeda, P. McElligott, K. Ryan, and S. Thiel, "Model-based testing design for embedded automotive software," SAE Technical Paper 2009-01-0151 2009.

[Source 175] J. Zander, "Model-based testing for execution algorithms in the simulation of cyber-physical systems," IEEE Systems Readiness Technology Conference (AUTOTESTCON), , 2013, pp. 1-7.

[Source 176] P. Skruch, M. Panek, and B. Kowalczyk, "Model-based testing in embedded automotive systems," Model-Based Testing for Embedded Systems, CRC Press, pp. 293-308, 2011.

[Source 177] K. Lamberg, M. Beine, M. Eschmann, R. Otterbach, M. Conrad, and I. Fey, "Model-based testing of embedded automotive software using MTest," SAE Technical Paper 0148-7191, 2004.

[Source 178] J. Keränen and T. Räty, "Model-based testing of embedded systems in hardware in the loop environment," IET Software , vol. 6, pp. 364-376, 2012.

[Source 179] D. Streitferdt, F. Kantz, P. Nenninger, T. Ruschival, H. Kaul, T. Bauer, et al., "Model-based testing of highly configurable embedded systems in the automation domain," Adoption and Optimization of Embedded and Real-Time Communication Systems, eIGI Global, 2013, pp. 108-125.

[Source 180] J. Zander-Nowicka, "Model-based testing of real-time embedded systems in the automotive domain", Doctoral Thesis, Technical University of Berlin 2009.

[Source 181] H. Shokry and M. Hinchey, "Model-based verification of embedded software", IEEE Computer, vol. 42, no. 4, pp. 53-59, 2009.

[Source 182] J. Großmann, P. Makedonski, H.-W. Wiesbrock, J. Svacina, I. Schieferdecker, and J. Grabowski, "Model-based X-in-the-loop testing," Model-Based Testing for Embedded Systems, CRC Press, pp. 299-338, 2011.

[Source 183] W. Yichen, L. Xinsheng, and W. Yikun, "Modeling embedded software test requirement based on MARTE," IEEE International Conference on Software Security and Reliability-Companion, 2013, pp. 109-115.

[Source 184] J. McDonald, L. Murray, P. Lindsay, and P. Strooper, "Module testing embedded software-an industrial pilot project," Proceedings of IEEE International Conference on Engineering of Complex Computer Systems, 2001, pp. 233-238.

[Source 185] K. P. Bulenkov, "MotoCAP STA Framework: New approach to testing of embedded systems," IEEE International Symposium on Consumer Electronics, 2007, pp. 1-2.

[Source 186] T. Tite, A. Vig, N. Olteanu, and C. Cuna, "moviTest: A Test Environment dedicated to multi-core embedded architectures," International Symposium on System on Chip, 2011, pp. 108-111.

[Source 187] S. Schulz, K. J. Buchenrieder, and J. W. Rozenblit, "Multilevel testing for design verification of embedded systems," IEEE Design & Test of Computers , vol. 19, pp. 60-69, 2002.

[Source 188] A. M. Pérez and S. Kaiser, "Multilevel Testing for Embedded Systems," Model-Based Testing for Embedded Systems, CRC Press, pp. 269, 2011.

[Source 189] S. Petters, D. Thomas, M. Friedmann, and O. Von Stryk, "Multilevel testing of control software for teams of autonomous mobile robots," Simulation, Modeling, and Programming for Autonomous Robots , pp. 183-194, 2008.

[Source 190] R. Özkan, V. Garousi, and A. Betin-Can, "Multi-objective regression test selection in practice: an empirical study in the defense software industry," in Proceedings of ACM/IEEE International Symposium on Empirical Software Engineering and Measurement , 2017.

[Source 191] A. Sung, B. Choi, W. E. Wong, and V. Debroy, "Mutant generation for embedded systems using kernel-based software and hardware fault simulation," Information and Software Technology , vol. 53, pp. 1153-1164, 2011.

[Source 192] E. P. Enoiu, D. Sundmark, A. Čaušević, R. Feldt, and P. Pettersson, "Mutation-Based Test Generation for PLC Embedded Software Using Model Checking," in IFIP International Conference on Testing Software and Systems, 2016, pp. 155-171.

[Source 193] S. P. Karmore and A. R. Mahajan, "New Approach for Testing and providing Security Mechanism for Embedded Systems," Procedia Computer Science , vol. 78, pp. 851-858, 2016.

[Source 194] C. Hote, "Next generation testing technique for embedded software: abstract semantics analysis," in Digital Avionics Systems Conference, 2004, pp. 10

[Source 195] P. Graf, K. D. Muller-Glaser, and C. Reichmann, "Nonintrusive black-and white-box testing of embedded systems software against UML Models," in IEEE/IFIP International Workshop on Rapid System Prototyping, 2007, pp. 130-138.

[Source 196] B. O. Obele and D. Kim, "On an Embedded Software Design Architecture for Improving the Testability of In-vehicle Multimedia Software," in Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on, 2014, pp. 349-352.

[Source 197] Y. Yongfeng, L. Bin, and Z. Bentao, "On test script technique oriented automation of embedded software simulation testing," in Computer Science and Information Engineering, 2009 WRI World Congress on, 2009, pp. 727-732.

[Source 198] C. Pfaller, A. Fleischmann, J. Hartmann, M. Rappl, S. Rittmann, and D. Wild, "On the integration of design and test: a model-based approach for embedded systems," in Proceedings of international workshop on Automation of software test, 2006, pp. 15-21.

[Source 199] G. Di Guglielmo, L. Di Guglielmo, A. Foltinek, M. Fujita, F. Fummi, C. Marconcini, et al., "On the integration of model-driven design and dynamic assertion-based verification for embedded software," Journal of Systems and Software , vol. 86, 2013.

[Source 200] K. Alnawasreh, P. Pelliccione, Z. Hao, M. Rånge, and A. Bertolino, "Online robustness testing of distributed embedded systems: an industrial approach," in Proceedings of International Conference on Software Engineering: Software Engineering in Practice Track, 2017, pp. 133-142.

[Source 201] J. Behrend, A. Gruenhage, D. Schroeder, D. Lettnin, J. Ruf, T. Kropf, et al., "Optimized hybrid verification of embedded software," in Latin American Test Workshop, 2014, pp. 1-6.

[Source 202] D. You, I. Amundson, S. A. Hareland, and S. Rayadurgam, "Practical aspects of building a constrained random test framework for safety-critical embedded systems," in Proceedings of International Workshop on Modern Software Engineering Methods for Industrial Automation, 2014, pp. 17-25.

[Source 203] J. Straunstrup, H. R. Andersen, H. Hulgaard, J. Lind-Nielsen, G. Behrmann, K. Kristoffersen, et al., "Practical verification of embedded software," IEEE Computer , vol. 33, pp. 68-75, 2000.

[Source 204] J. Pesonen, M. Katara, and T. Mikkonen, "Production-testing of embedded systems with aspects," in Haifa Verification Conference, 2005, pp. 90-102.

[Source 205] M. Kammerstetter, C. Platzer, and W. Kastner, "Prospect: peripheral proxying supported embedded code testing," in Proceedings of ACM symposium on Information, computer and communications security, 2014, pp. 329-340.

[Source 206] J. Zander-Nowicka, P. J. Mosterman, and I. Schieferdecker, "Quality of test specification by application of patterns," in Proceedings of Conference on Pattern Languages of Programs, 2008, p. 3.

[Source 207] W.-T. Tsai, L. Yu, F. Zhu, and R. Paul, "Rapid embedded system testing using verification patterns," IEEE software , vol. 22, pp. 68-75, 2005.

[Source 208] J. Vain, A. Kull, M. Kääramees, M. Markvardt, and K. Raiend, "Reactive Testing of Nondeterministic Systems by Test Purpose-Directed Tester," Model-Based Testing for Embedded Systems, CRC Press, pp. 425-452, 2011.

[Source 209] V. Gafni, R. Shelef, and H. Tibber, "A real-time simulation environment for embedded computer systems software testing," in Proceedings of Israel Conference on Computer Systems and Software Engineering, 1989, pp. 16-21.

[Source 210] F. Sun and Y. Li, "Regression testing Prioritization Based on model Checking for Safety-Crucial Embedded Systems," in International Conference on Digital Manufacturing and Automation, 2013, pp. 979-983.

[Source 211] Z. Qin, H. Chen, and Y. Shi, "Reliability demonstration testing method for safety-critical embedded applications software," in International Conference on Embedded Software and Systems, 2008, pp. 481-487.

[Source 212] W. Liu and B. Liu, "Research and Design on Real-Time Embedded Software Simulation Testing Environment," Advances in Biomedical Engineering , vol. 8, p. 366, 2012.

[Source 213] X.-l. Lu, Y.-w. Dong, B. Sun, and H.-b. Zhao, "Research of embedded software testing method based on AADL modes," in IEEE International Conference on Communication Software and Networks, 2011, pp. 89-92.

[Source 214] J. Yin, J. Zheng, H. Teng, and F. Li, "Research of testing technology for gun's initial disturbance based on embedded PC104 and PSD optical lever," in International Conference on Electronic Measurement & Instruments, 2009, pp. 1-923-1-927.

[Source 215] D. Ting and W. Zhongmin, "Research on instrumentation technology for the embedded assembly software testing based on host computer," in International Conference on Computer Science and Network Technology, 2011, pp. 1906-1909.

[Source 216] J. Kloos, T. Hussain, and R. Eschbach, "Risk-based testing of safety-critical embedded systems driven by fault tree analysis," in IEEE International Conference on Software Testing, Verification and Validation, 2011, pp. 26-33.

[Source 217] S. M. A. Shah, D. Sundmark, B. Lindström, and S. F. Andler, "Robustness testing of embedded software systems: An industrial interview study," IEEE Access , vol. 4, pp. 1859-1871, 2016.

[Source 218] W.-T. Tsai, L. Yu, X. Liu, A. Saimi, and Y. Xiao, "Scenario-based test case generation for state-based embedded systems," in Proceedings of IEEE International Performance, Computing, and Communications Conference, 2003, pp. 335-342.

[Source 219] Y. Kim and M. Kim, "SCORE: a scalable concolic testing tool for reliable embedded software," in Proceedings of ACM SIGSOFT symposium and European conference on Foundations of software engineering, 2011, pp. 420-423.

[Source 220] R. Matinnejad, S. Nejati, L. Briand, T. Bruckmann, and C. Poull, "Search-based automated testing of continuous controllers: Framework, tool support, and case studies," Information and Software Technology , vol. 57, pp. 705-722, 2015.

[Source 221] K. Doganay, S. Eldh, W. Afzal, and M. Bohlin, "Search-based testing for embedded telecommunication software with complex input structures: An industrial case study," Swedish Institute of Computer Science, technical report T2014:03, 2014.

[Source 222] K. Doganay, M. Bohlin, and O. Sellin, "Search based testing of embedded systems implemented in IEC 61131-3: An industrial case study," in IEEE International Conference on Software Testing, Verification and Validation, 2013, pp. 425-432.

[Source 223] J. Wegener and P. M. Kruse, "Search-based testing with in-the-loop systems," in International Symposium on Search-Based Software Engineering, 2009, pp. 81-84.

[Source 224] V. Santiago, W. P. Silva, and N. L. Vijaykumar, "Shortening test case execution time for embedded software," in International Conference on Secure System Integration and Reliability Improvement, 2008, pp. 81-88.

[Source 225] J. Wegener and P. M. Kruse, "Search-based testing with in-the-loop systems," in International Symposium on Search Based Software Engineering, 2009, pp. 81-84.

[Source 226] T. Yu, W. Srisa-an, and G. Rothermel, "SimTester: a controllable and observable testing framework for embedded systems," in ACM SIGPLAN Notices, 2012, pp. 51-62.

[Source 227] P. Grillinger, P. Brada, and S. Racek, "Simulation approach to embedded system programming and testing," in IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, 2004, pp. 248-254.

[Source 228] Y. Wang and Z. Zhou, "Software BIT design and testing for embedded software," in Reliability, Maintainability and Safety, 2009. International Conference on Reliability, Maintainability and Safety, 2009, pp. 703-707.

[Source 229] A. A. Samuel, N. Jayalal, B. Valsa, C. Ignatious, and J. P. Zachariah, "Software Fault Injection Testing of the Embedded Software of a Satellite Launch Vehicle," IEEE Potentials , vol. 32, pp. 38-44, 2013.

[Source 230] M. Muresan and D. Pitica, "Software in the Loop environment reliability for testing embedded code," in IEEE International Symposium for Design and Technology in Electronic Packaging, 2012, pp. 325-328.

[Source 231] D. K. Saini, "Software testing for embedded systems," International Journal of Computer Applications , vol. 43, 2012.

[Source 232] P. Braione, G. Denaro, A. Mattavelli, M. Vivanti, and A. Muhammad, "Software testing with code-based test generators: data and lessons learned from a case study with an industrial software component," Software quality Journal , vol. 22, pp. 311-333, 2014.

[Source 233] C. M. Kirchsteiger, C. Trummer, C. Steger, R. Weiss, and M. Pistauer, "Specification-based Verification of Embedded Systems by Automated Test Case Generation," Working Conference on Distributed and Parallel Embedded Systems, 2008, pp. 35-44.

[Source 234] A. Mjeda, "Standard-compliant testing for safety-related automotive software," PhD thesis, University of Limerick, 2013.

[Source 235] C.-S. Koong, H.-J. Lai, C.-H. Chang, W. C. Chu, N.-L. Hsueh, P.-A. Hsiung, et al., "Supporting tool for embedded software testing," International Conference on Quality Software, 2010, pp. 481-487.

[Source 236] S. Chakrabarti and S. Ramesh, "Symtest: A framework for symbolic testing of embedded software," in Proceedings of India Software Engineering Conference, 2016, pp. 48-58.

[Source 237] M. Conrad, I. Fey, and S. Sadeghipour, "Systematic model-based testing of embedded automotive software," Electronic Notes in Theoretical Computer Science , vol. 111, pp. 13-26, 2005.

[Source 238] A. Krupp and W. Müller, "Systematic Model-in-the-Loop Test of Embedded Control Systems," in International Embedded Systems Symposium, 2009, pp. 171-184.

[Source 239] J. Zander-Nowicka, X. Xiong, and I. Schieferdecker, "Systematic Test Data Generation for Embedded Software," in Software Engineering Research and Practice, 2008, pp. 164-170.

[Source 240] M. Conrad, "Systematic testing of embedded automotive software-the classification-tree method for embedded systems (CTM/ES)," in Dagstuhl Seminar Proceedings, 2005.

[Source 241] A. Fin, F. Fummi, M. Martignano, and M. Signoretto, "SystemC: a homogenous environment to test embedded systems," Proceedings of International Symposium on Hardware/Software Co-design, 2001, pp. 17-22.

[Source 242] N. Van Schooenderwoert and R. Morsicato, "Taming the embedded tiger-agile test techniques for embedded software," in Agile Development Conference, 2004, pp. 120-126.

[Source 243] S. Biswas, R. Mall, and M. Satpathy, "Task dependency analysis for regression test selection of embedded programs," IEEE Embedded Systems Letters , vol. 3, pp. 117-120, 2011.

[Source 244] S. S. Bhattacharyya, W. Plishker, A. Gupta, and C.-C. Shen, "Teaching cross-platform design and testing methods for embedded systems using DICE," in Proceedings of Workshop on Embedded Systems Education, 2011, pp. 38-45.

[Source 245] W. Fleisch, "Test Case Design for the Validation of Component-Based Embedded Systems," in Architecture and Design of Distributed Embedded Systems, Springer, 2001, pp. 151-160.

[Source 246] C. Schwarzl and F. Wotawa, "Test case generation in practice for communicating embedded systems," Electrical and Computer Engineering , vol. 128, pp. 240-244, 2011.

[Source 247] Y. Yongfeng, L. Bin, L. Minyan, and L. Zhen, "Test cases generation for embedded real-time software based on extended UML," International Conference on Information Technology and Computer Science, 2009, pp. 69-74.

[Source 248] S. P. Masticola and M. Gall, "Test Framework Architectures for Model-Based Embedded System Testing," Model-Based Testing for Embedded Systems, CRC Press, pp. 49, 2011.

[Source 249] T. Chen, X.-S. Zhang, X.-L. Ji, C. Zhu, Y. Bai, and Y. Wu, "Test Generation for Embedded Executables via Concolic Execution in a Real Environment," IEEE Transactions on Reliability , vol. 64, pp. 284-296, 2015.

[Source 250] F. Dadeau, F. Peureux, B. Legeard, R. Tissot, J. Julliand, P.-A. Masson, et al., "Test generation using symbolic animation of models," Model-Based Testing for Embedded Systems, CRC Press, pp. 195-218, 2011.

[Source 251] T. Reinbacher, J. Brauer, M. Horauer, A. Steininger, and S. Kowalewski, "Test-case generation for embedded binary code using abstract interpretation", Dagstuhl-Leibniz-Center for Computer Science, 2011.

[Source 252] N. He, P. Rümmer, and D. Kroening, "Test-case generation for embedded simulink via formal concept analysis," in Proceedings of Design Automation Conference, 2011, pp. 224-229.

[Source 253] P. Cordemans, S. Van Landschoot, J. Boydens, and E. Steegmans, "Test-Driven Development as a Reliable Embedded Software Engineering Practice," in Embedded and Real Time System Development: A Software Engineering Perspective, Springer, 2014, pp. 91-130.

[Source 254] J. Boydens, P. Cordemans, and E. Steegmans, "Test-driven development of embedded software," in Proceedings of the Fourth European Conference on the Use of Modern Information and Communication Technologies, 2010, pp. 117-128.

[Source 255] P. Cordemans, J. Boydens, S. Van Landschoot, and E. Steegmans, "Test-driven development strategies applied to embedded software," in Proceedings of the European conference on the use of modern information and communication technologies, 2012, pp. 31-42.

[Source 256] H. Chae, X. Jin, S. Lee, and J. Cho, "TEST: testing environment for embedded systems based on TTCN-3 in SILS," Advances in Software Engineering , pp. 204-212, 2009.

[Source 257] G. Tibba, C. Malz, C. Stoermer, N. Nagarajan, L. Zhang, and S. Chakraborty, "Testing automotive embedded systems under X-in-the-loop setups," in Proceedings of International Conference on Computer-Aided Design, 2016, p. 35.

[Source 258] B. Kim, Y. Kashiba, S. Dai, and S. Shiraishi, "Testing Autonomous Vehicle Software in the Virtual Prototyping Environment," IEEE Embedded Systems Letters , vol. 9, pp. 5-8, 2017.

[Source 259] T. Arts, J. Hughes, U. Norell, and H. Svensson, "Testing AUTOSAR software with QuickCheck," in IEEE International Conference on Software Testing, Verification and Validation, 2015, pp. 1-4.

[Source 260] J. Hunt, "Testing control software using a genetic algorithm," Engineering Applications of Artificial Intelligence , vol. 8, pp. 671-680, 1995.

[Source 261] D. S. Loubach, J. C. Nobre, A. M. Da Cunha, L. A. Dias, M. R. Nascimento, and W. A. Dos Santos, "Testing critical software: A case study for an aerospace application," in IEEE/AIAA Digital Avionics Systems Conference, 2006, pp. 1-9.

[Source 262] I. Schieferdecker and J. Großmann, "Testing embedded control systems with ttcn-3," Software Technologies for Embedded and Ubiquitous Systems , pp. 125-136, 2007.

[Source 263] J. Grossmann, D. Serbanescu, and I. Schieferdecker, "Testing embedded real time systems with TTCN-3," in International Conference on Software Testing Verification and Validation, 2009, pp. 81-90.

[Source 264] F.-C. Kuo, T. Y. Chen, and W. K. Tam, "Testing embedded software by metamorphic testing: A wireless metering system case study," IEEE Conference on Local Computer Networks , 2011, pp. 291-294.

[Source 265] J. Engblom, G. Girard, and B. Werner, "Testing Embedded Software using Simulated Hardware," European Congress on Embedded Real Time Software , pp. 1-9, 2006.

[Source 266] T. Yu, "Testing Embedded System Applications", PhD thesis, University of Nebraska, 2010.

[Source 267] T. Kuroiwa and N. Kushiro, "Testing environment for embedded software product lines," IEEE/ACS International Conference of Computer Systems and Applications , 2015, pp. 1-7.

[Source 268] F. Saglietti, "Testing for dependable embedded software," EUROMICRO Conference on Software Engineering and Advanced Applications, 2010, pp. 409-416.

[Source 269] J. Großmann, "Testing hybrid systems with TTCN-3 embedded", International Journal on Software Tools for Technology Transfer, Volume 16, Issue 3, pp. 247–267, 2014.

[Source 270] M. Oriol, "Testing legacy embedded code: Landing on a software engineering desert island," in IEEE International Conference on Software Testing, Verification and Validation, 2015, pp. 1-2.

[Source 271] Z. Havlice, J. Vizi, and V. Szaboova, "Testing medical embedded software," in IEEE International Symposium on Applied Machine Intelligence and Informatics, 2014, pp. 99-102.

[Source 272] P. Sampath, A. Rajeev, S. Ramesh, and K. Shashidhar, "Testing model-processing tools for embedded systems," in IEEE Real Time and Embedded Technology and Applications Symposium , 2007, pp. 203-214.

[Source 273] S. Cha, S. Jeong, J. Yoo, and Y.-G. Kim, "Testing of safety-critical software embedded in an artificial heart," Advances in Systems Safety , pp. 143-153, 2011.

[Source 274] Y. Moy, E. Ledinot, H. Delseny, V. Wiels, and B. Monate, "Testing or formal verification: Do-178c alternatives and industrial experience," IEEE software , vol. 30, pp. 50-57, 2013.

[Source 275] M. Jamro and B. Trybus, "Testing procedure for IEC 61131-3 control software," Proceedings of International Federation of Automatic Control Conference , vol. 46, pp. 192-197, 2013.

[Source 276] W. Pingping, Q. Xiaoyao, and Z. Jiang, "Testing process model and classification of test methods for embedded software of electrical products," in IEEE International Conference on Software Engineering and Service Science, 2012, pp. 204-207.

[Source 277] L. T. Beque, T. Dai Pra, and E. Cota, "Testing requirements for an embedded operating system: The exception handling case study," in Latin American Test Workshop, 2009, pp. 1-6.

[Source 278] M. Ramachandran, "Testing reusable software components from object specification," ACM SIGSOFT Software Engineering Notes , vol. 28, p. 18, 2003.

[Source 279] H. Y. Kim and F. T. Sheldon, "Testing Software Requirements with Z and Statecharts Applied to an Embedded Control Systemt0t1," Software Quality Journal , vol. 12, pp. 231-264, 2004.

[Source 280] Y. Shunkun, T. Dongxiao, and S. Xiaohua, "Testing System for CAN Bus-oriented Embedded Software," IEEE/ACIS International Conference on Computer and Information Science, 2014.

[Source 281] J. L. Davidson and G.-A. Amoussou, "Testing to certify an embedded software system," Journal of Computing Sciences in Colleges , vol. 25, pp. 83-90, 2010.

[Source 282] H.-L. Truong and L. Berardinelli, "Testing uncertainty of cyber-physical systems in IoT cloud infrastructures: combining model-driven engineering and elastic execution," in Proceedings of ACM SIGSOFT International Workshop on Testing Embedded and Cyber-Physical Systems, 2017, pp. 5-8.

[Source 283] F. Kantz, T. Ruschival, P. Nenninger, and D. Streitferdt, "Testing with large parameter sets for the development of embedded systems in the automation domain," in IEEE International Computer Software and Applications Conference , 2009, pp. 504-509.

[Source 284] M. Conrad, "Testing-Based Translation Validation of Generated Code," Model-Based Testing for Embedded Systems, CRC Press, , pp. 579-599, 2011.

[Source 285] J. Grossmann, I. Fey, A. Krupp, M. Conrad, C. Wewetzer, and W. Mueller, "Testml-a test exchange language for model-based testing of embedded software," in Automotive Software Workshop, 2006, pp. 98-117.

[Source 286] L. Fan, Z. Wenhua, C. Guowu, and J. Yi, "The embedded product testing using cleanroom statistical method," in World Congress on Computer Science and Information Engineering, 2009, pp. 750-754.

[Source 287] W. Ping, "The fault-tolerant design and fault injection test for embedded software," in Pacific-Asia Conference on Circuits, Communications and System, 2010, pp. 307-310.

[Source 288] Y. Yin, B. Liu, Z. Li, C. Zhang, and N. Wu, "The Integrated Application Based on Real-time Extended UML and Improved Formal Method in Real-time Embedded Software Testing," Journal of Communications and Networks , vol. 5, pp. 1410-1416, 2010.

[Source 289] Z. Wei, L. Qi, and Z. Liang, "The research of statistical testing fault detection method based on embedded software," in International Conference on New Trends in Information Science and Service Science and Data Mining, 2012, pp. 120-124.

[Source 290] L. Shuping and P. Ling, "The research of V model in testing embedded software," in International Conference on Computer Science and Information Technology, 2008, pp. 463-466.

[Source 291] L. Xiao, M. Gu, and J. Sun, "The testing approach of embedded real-time automatic control software based on control objects," in World Congress on Intelligent Control and Automation, 2008, pp. 4178-4183.

[Source 292] Y. Wang, Y. Wang, and C. Jiang, "Theory and implemetion of Simulation Testing Framework for embedded software," in International Conference on Reliability, Maintainability and Safety, 2011, pp. 751-756.

[Source 293] P. Iyenghar, E. Pulvermueller, M. Spieker, J. Wuebbelmann, and C. Westerkamp, "Time and memory-aware runtime monitoring for executing model-based test cases in embedded systems," in IEEE International Conference on Industrial Informatics, 2013, pp. 506-512.

[Source 294] C. Wiederseiner, V. Garousi, and M. Smith, "Tool support for automated traceability of test/code artifacts in embedded software systems," in IEEE International Conference on, Trust, Security and Privacy in Computing and Communications2011, pp. 1109-1117.

[Source 295] Y. Ki, J. Seo, B. Choi, and K. La, "Tool support for new test criteria on embedded systems: Justitia," in Proceedings of international conference on Ubiquitous information management and communication, 2008, pp. 365-369.

[Source 296] D. Amalfitano, A. R. Fasolino, S. Scala, and P. Tramontana, "Towards automatic model-in-the-loop testing of electronic vehicle information centers," in Proceedings of international workshop on Long-term industrial collaboration on software engineering, 2014, pp. 9-12.

[Source 297] Y. Jiao, K. Zhu, Q. Yu, and B. Wu, "Towards model-driven methodology: a novel testing approach for collaborative embedded system design," in International Conference on Computer Supported Cooperative Work in Design, 2006, pp. 1-5.

[Source 298] S. Marrone, F. Flammini, N. Mazzocca, R. Nardone, and V. Vittorini, "Towards Model-Driven V&V assessment of railway control systems," International Journal on Software Tools for Technology Transfer , vol. 16, pp. 669-683, 2014.

[Source 299] H. Koehnemann and T. Lindquist, "Towards target-level testing and debugging tools for embedded software," in Proceedings of the conference on TRI-Ada, 1993, pp. 288-298.

[Source 300] S. Vöst and S. Wagner, "Trace-based test selection to support continuous integration in the automotive industry," in Proceedings of International Workshop on Continuous Software Evolution and Delivery, 2016, pp. 34-40.

[Source 301] S. Salvi, D. Kästner, T. Bienmüller, and C. Ferdinand, "True Error or False Alarm? Refining Astrée's Abstract Interpretation Results by Embedded Tester's Automatic Model-Based Testing," in International Conference on Computer Safety, Reliability, and Security, 2014, pp. 84-96.

[Source 302] S. Blom, T. Deiß, N. Ioustinova, A. Kontio, J. Van De Pol, A. Rennoch, et al., "TTCN-3 for distributed testing embedded software," in International Andrei Ershov Memorial Conference on Perspectives of System Informatics, 2006, pp. 98-111.

[Source 303] F. Erculiani, L. Abeni, and L. Palopoli, "uBuild: Automated Testing and Performance Evaluation of Embedded Linux Systems," in International Conference on Architecture of Computing Systems, 2014, pp. 123-134.

[Source 304] J. Metsä, S. Maoz, M. Katara, and T. Mikkonen, "Using aspects for testing of embedded software: experiences from two industrial case studies," Software Quality Journal , vol. 22, pp. 185-213, 2014.

[Source 305] C. G. Bernardo, D. A. Montini, D. D. Fernandes, D. A. da Silva, L. A. V. Dias, and A. M. da Cunha, "Using GQM for testing design patterns in real-time and embedded systems on a software production line," in International Conference on Information Technology: New Generations, 2009, pp. 1397-1404.

[Source 306] T. Yu, A. Sung, W. Srisa-an, and G. Rothermel, "Using property-based oracles when testing embedded system applications," in IEEE International Conference on Software Testing, Verification and Validation, 2011, pp. 100-109.

[Source 307] H. Yu, H. Song, L. Xiaoming, and Y. Xiushan, "Using symbolic execution in embedded software testing," in International Conference on Computer Science and Software Engineering, 2008, pp. 738-742.

[Source 308] G. Sagardui, L. Etxeberria, and J. Agirre, "Variability management in testing architectures for embedded control systems," in International Conference on Advances in System Testing and Validation Lifecycle, 2012, pp. 73-78.

[Source 309] S.-K. Kim, J. Choi, D. Lee, S. H. Noh, and S. L. Min, "Virtual framework for testing the reliability of system software on embedded systems," in Proceedings of ACM symposium on Applied computing, 2007, pp. 1192-1196.

[Source 310] M. Becker, D. Baldin, C. Kuznik, M. M. Joy, T. Xie, and W. Mueller, "XEMU: an efficient QEMU based binary mutation testing framework for embedded software," in Proceedings of ACM international conference on Embedded software, 2012, pp. 33-42.

[Source 311] M. H. Netkow and D. Brylow, "Xest: an automated framework for regression testing of embedded software," in Proceedings of Workshop on Embedded Systems Education, 2010, p. 7.

[Source 312] J. Bräuer, H. Kleinwechter, and A. Leicher, "µTTCN-an approach to continuous signals in TTCN-3," in Software Engineering (Workshops), 2007, pp. 55-64.

8.2 OTHER REFERENCES

[1] S. Schulz, K. J. Buchenrieder, and J. W. Rozenblit, "Multilevel testing for design verification of embedded systems," *Design & Test of Computers, IEEE*, vol. 19, pp. 60-69, 2002.

[2] C. Ebert and C. Jones, "Embedded Software: Facts, Figures, and Future," *IEEE Computer*, vol. 42, pp. 42-52, 2009.

[3] V. Garousi, "A Systematic Approach to Software Test Automation and How to Increase its ROI," *Invited Talk, TestIstanbul industry conference, Istanbul, Turkey*, May 2013.

[4] V. Garousi and K. Herkiloğlu, "Selecting the right topics for industry-academia collaborations in software testing: an experience report," in *IEEE International Conference on Software Testing, Verification, and Validation*, 2016, pp. 213-222.

[5] V. Garousi, M. M. Eskandar, and K. Herkiloğlu, "Industry-academia collaborations in software testing: experience and success stories from Canada and Turkey," *Software Quality Journal, special issue on Industry Academia Collaborations in Software Testing*, pp. 1-53, 2016.

[6] V. Garousi, "Career paths, training and learning resources in software test engineering," *Invited Talk, 'Ankara Testing Days-7' industry conference, Ankara, Turkey*, May 2016.

[7] V. Garousi, A. Coşkunçay, A. B. Can, and O. Demirörs, "A survey of software engineering practices in Turkey," *Journal of Systems and Software*, vol. 108, pp. 148-177, 2015.

[8] V. Garousi and J. Zhi, "A survey of software testing practices in Canada," *Journal of Systems and Software*, vol. 86, pp. 1354-1376, 2013.

[9] G. Urul, V. Garousi, and G. Urul, "Test Automation for Embedded Real-time Software: An Approach and Experience Report in the Turkish Industry," in *Proceedings of the Turkish National Software Engineering Symposium "Ulusal Yazılım Mühendisliği Sempozyumu" (UYMS)*, 2014.

[10] A. Banerjee, S. Chattopadhyay, and A. Roychoudhury, "On Testing Embedded Software," *Advances in Computers*, vol. 101, pp. 121–153, 2016.

[11] M. A. J. Burford and F. Belli, "CADAS: A Tool for Designing Reliable Embedded Software and Supporting Testing "in the Large"," in *Fehlertolerierende Rechensysteme*. vol. 84, K. E. Großpietsch and M. Dal Cin, Eds., ed: Springer Berlin Heidelberg, 1984, pp. 101-112.

[12] T. Dyba and T. Dingsoyr, "What Do We Know about Agile Software Development?," *IEEE Software*, vol. 26, pp. 6-9, 2009.

[13] T. Hall, H. Sharp, S. Beecham, N. Baddoo, and H. Robinson, "What Do We Know about Developer Motivation?," *IEEE Software*, vol. 25, pp. 92-94, 2008.

[14] Auriga Corp., "Testing of Embedded Software Products," in *https://www.auriga.com/files/projects/Auriga_test_of_emb_sw_prod.pdf*, Last accessed: April 2017.

[15] Kualitatem Corp., "Embedded Software Testing Challenges," in *https://www.kualitatem.com/blog/embedded-software-testing-challenges*, Last accessed: April 2017.

[16] V. Garousi and M. V. Mäntylä, "A systematic literature review of literature reviews in software testing," *Information and Software Technology*, vol. 80, pp. 195-216, 2016.

[17] V. Garousi, "Online Paper Repository for Systematic Mapping of Secondary studies in software testing," in *http://goo.gl/Oxb0X8*, Last accessed: Sept. 2015.

[18] W. Afzal, R. Torkar, and R. Feldt, "A systematic mapping study on non-functional search-based software testing," in *International Conference on Software Engineering and Knowledge Engineering*, 2008, pp. 488-493.

[19] P. A. d. M. S. Neto, I. d. C. Machado, J. D. McGregord, E. S. d. Almeida, and S. R. d. L. Meira, "A systematic mapping study of software product lines testing," *Information and Software Technology*, vol. 53, pp. 407 - 423, 2011.

[20] I. Banerjee, B. Nguyen, V. Garousi, and A. Memon, "Graphical User Interface (GUI) Testing: Systematic Mapping and Repository," *Information and Software Technology*, vol. 55, pp. 1679–1694, 2013.

[21] Ç. Çatal and D. Mishra, "Test Case Prioritization: A systematic mapping Study " *Software Quality Journal*, vol. 21, pp. 445-478, 2013.

[22] V. G. Yusifoğlu, Y. Amannejad, and A. Betin-Can, "Software Test-Code Engineering: A Systematic Mapping," *Journal of Information and Software Technology*, In Press, 2014.

[23] A. C. D. Neto, R. Subramanyan, M. Vieira, and G. H. Travassos, "A Survey on Model-based Testing Approaches- A systematic review," in *Proceedings of the ACM International Workshop on Empirical Assessment of Eoftware Engineering languages and technologies*, 2007.

[24] B. Haugset and G. K. Hanssen, "Automated Acceptance Testing-A Literature Review and an Industrial Case Study," in *Agile Conference*, 2008, pp. 27 - 38.

[25] P. K. Singh, O. P. Sangwan, and A. Sharma, "A systematic review on fault-based mutation testing techniques and tools for Aspect-J programs," in *IEEE International Advance Computing Conference*, 2013, pp. 1455 - 1461.

[26] S. Doğan, A. Betin-Can, and V. Garousi, "Web application testing: A systematic literature reviewr," *Journal of Systems and Software* vol. 91, pp. 174–201, 2014.

[27] U. Kanewala and J. M. Bieman, "Testing scientific software: A systematic literature review," *Information and Software Technology,* vol. 56, pp. 1219-1232, 10// 2014.

[28] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines-a survey," *Proceedings of the IEEE,* vol. 84, pp. 1090 - 1123, 1996.

[29] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Softw. Test. Verif. Reliab.,* vol. 22, pp. 67-120, 2012.

[30] M. Bozkurt, M. Harman, and Y. Hassoun, "Testing and Verification In Service-Oriented Architecture-A Survey," *Journal of Software Testing, Verification and Reliability,* vol. 23, pp. 261-313, 2013.

[31] M. Shirole and R. Kumar, "UML behavioral model based test case generation: a survey," *ACM SIGSOFT Software Engineering Notes,* vol. 38, pp. 1-13, 2013.

[32] M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "A comprehensive Survey of Trends in Oracles for Software Testing," *In Press, IEEE Transactions on Software Engineering,* 2014.

[33] B. Kitchenham, P. Brereton, and D. Budgen, "The educational value of mapping studies of software engineering literature," in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, 2010, pp. 589-598.

[34] R. W. Schlosser, "The Role of Systematic Reviews in Evidence-Based Practice, Research, and Development," *FOCUS Technical Brief #15, http://ktdrr.org/ktlibrary/articles_pubs/ncddrwork/focus/focus15/Focus15.pdf*, 2006.

[35] M. Bell, P. Cordingley, C. Isham., and R. Davis., "Report of Professional Practitioner Use of Research Review: Practitioner engagement in and/or with research," *Coventry: CUREE, GTCE, LSIS & NTRP. Available at: http://www.curee-paccts.com/node/2303*, 2010.

[36] H. Aveyard, *Doing A Literature Review In Health And Social Care: A Practical Guide: A Practical Guide Paperback*, 2 edition ed.: Open University Press, 2010.

[37] J. R. Barbosa, A. M. R. Vincenzi, M. E. Delamaro, and J. C. Maldonado, "Software Testing in Critical Embedded Systems: a Systematic Review of Adherence to the DO-178B Standard," in *The Third International Conference on Advances in System Testing and Validation Lifecycle*, 2011.

[38] M. Johnson, "In, but not of, the system: overview of embedded systems test methods," in *Northcon 95. I EEE Technical Applications Conference and Workshops Northcon95*, 1995, p. 36.

[39] J. S. Keranen and T. D. Raty, "Model-based testing of embedded systems in hardware in the loop environment," *IET Software,* vol. 6, pp. 364-376, 2012.

[40] S. Liu, "Evaluation of model-based testing for embedded systems based on the example of the safety-critical vehicle functions," MSc, Department of Computer Science and Engineering, University of Gothenburg, 2012.

[41] S. Oster, A. Wübbeke, G. Engels, and A. Schürr, "A Survey of Model-Based Software Product Lines Testing," in *Model-Based Testing for Embedded Systems*, E. b. J. Zander, I. Schieferdecker, and P. J. Mosterman, Eds., ed: CRC Press, 2011.

[42] N. K. Bahrin and R. Mohamad, "A systematic literature review of test case generator for embedded real time system," *International Journal of Software Engineering and Technology,* vol. 1, pp. 38-45, 2014).

[43] E. Jahier, S. Djoko-Djoko, C. Maiza, and E. Lafont, "Environment-Model Based Testing of Control Systems: Case Studies," in *Tools and Algorithms for the Construction and Analysis of Systems: 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, E. Ábrahám and K. Havelund, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 636-650.

[44] J. Zhi, V. Garousi, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation: a systematic mapping," *Journal of Systems and Software,* vol. 99, pp. 175–198, 2015.

[45] V. Garousi, Y. Amannejad, and A. Betin-Can, "Software test-code engineering: a systematic mapping," *Journal of Information and Software Technology,* vol. 58, pp. 123–147, 2015.

[46] S. Doğan, A. Betin-Can, and V. Garousi, "Web Application Testing: A Systematic Literature Review," *Journal of Systems and Software,* vol. 91, pp. 174-201, 2014.

[47] F. Häser, M. Felderer, and R. Breu, "Software paradigms, assessment types and non-functional requirements in model-based integration testing: a systematic literature review," presented at the Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, 2014.

[48] M. Felderer, P. Zech, R. Breu, M. Büchler, and A. Pretschner, "Model-based security testing: a taxonomy and systematic classification," *Software Testing, Verification and Reliability,* 2015.

[49] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology,* vol. 64, pp. 1-18, 8// 2015.

[50] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," presented at the Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, United Kingdom, 2014.

[51] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008.

[52] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software engineering," 2007.

[53] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature," in *International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2016, pp. 171-176.

[54] V. Garousi, M. Felderer, and T. Hacaloğlu, "Software test maturity assessment and test process improvement: A multivocal literature review," *Information and Software Technology,* vol. 85, pp. 16–42, 2017.

[55] V. Garousi and M. V. Mäntylä, "When and what to automate in software testing? A multi-vocal literature review," *Information and Software Technology,* vol. 76, pp. 92-117, 2016.

[56] P. Ammann and J. Offutt, *Introduction to Software Testing*: Cambridge University Press, 2008.

[57] V. Garousi and M. V. Mäntylä, "Citations, research topics and active countries in software engineering: A bibliometrics study " *Elsevier Computer Science Review,* vol. 19, pp. 56-77, 2016.

[58] V. Garousi, "A bibliometric analysis of the Turkish software engineering research community," *Springer Journal on Scientometrics,* vol. 105, pp. 23-49, 2015.

[59] N. R. Haddaway, A. M. Collins, D. Coughlin, and S. Kirk, "The Role of Google Scholar in Evidence Reviews and Its Applicability to Grey Literature Searching," *PLoS ONE,* vol. 10, 2015.

[60] Multiple online users, "How are cyber-physical systems different from existing technologies like robotics, embedded systems, etc?," in *https://www.researchgate.net/post/How_are_cyber-physical_systems_different_from_existing_technologies_like_robotics_embedded_systems_etc*, Last accessed: April 2017.

[61] Multiple online users, "Is it fair and accurate to say that embedded systems is a subset of cyber physical systems?," in *https://www.quora.com/Is-it-fair-and-accurate-to-say-that-embedded-systems-is-a-subset-of-cyber-physical-systems*, Last accessed: April 2017.

[62] K. Godin, J. Stapleton, S. I. Kirkpatrick, R. M. Hanning, and S. T. Leatherdale, "Applying systematic review search methods to the grey literature: a case study examining guidelines for school-based breakfast programs in Canada," *Systematic Reviews,* vol. 4, pp. 138-148, Oct 22 2015.

[63] Q. Mahood, D. Van Eerd, and E. Irvin, "Searching for grey literature for systematic reviews: challenges and benefits," *Research Synthesis Methods,* vol. 5, pp. 221-234, 2014.

[64] J. Adams, F. C. Hillier-Brown, H. J. Moore, A. A. Lake, V. Araujo-Soares, M. White, *et al.*, "Searching and synthesising 'grey literature' and 'grey information' in public health: critical reflections on three case studies," *Systematic Reviews,* vol. 5, p. 164, 2016.

[65] V. Garousi, A. Mesbah, A. Betin-Can, and S. Mirshokraie, "A Systematic Mapping Study of Web Application Testing," *Elsevier Journal of Information and Software Technology,* vol. 55, pp. 1374–1396, 2013.

[66] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.,* vol. 45, pp. 1-61, 2012.

[67] Y. Jia and M. Harman, "An Analysis and Survey of the Development of Mutation Testing," *IEEE Transactions on Software Engineering,* vol. 37, pp. 649-678, 2011.

[68] R. V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*. New Jersey: Addison-Wesley, 2000.

[69] A. P. Mathur, *Foundations of Software Testing*: Addison-Wesley Professional, 2008.

[70] V. Garousi and D. Pfahl, "When to Automate Software Testing? A Decision-Support Approach Based on Process Simulation," *Wiley Journal of Software: Evolution and Process,* vol. 28, pp. 312-285, 2016.

[71] H. Shokry and M. Hinchey, "Model-Based Verification of Embedded Software," *Computer,* vol. 42, pp. 53-59, 2009.

[72] J. Day, "Virtual Hardware In the Loop (vHIL): Earlier and Better Testing for Automotive Applications," in *http://johndayautomotivelectronics.com/virtual-hardware-in-the-loop-earlier-and-better-testing-for-automotive-applications/*, Last accessed: April 2017.

[73] V. Garousi and G. Ruhe, "A Bibliometric/Geographic Assessment of 40 Years of Software Engineering Research (1969-2009)," *International Journal of Software Engineering and Knowledge Engineering,* vol. 23, pp. 1343–1366, 2013.

[74] R. Farhoodi, V. Garousi, D. Pfahl, and J. P. Sillito, "Development of Scientific Software: A Systematic Mapping, Bibliometrics Study and a Paper Repository," *International Journal of Software Engineering and Knowledge Engineering,* vol. 23, pp. 463-506, 2013.

[75] V. Garousi and J. M. Fernandes, "Highly-cited papers in software engineering: The top-100," *Information and Software Technology,* vol. 71, pp. 108-128, 2016.

[76] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*: Kluwer Academic Publishers, 2000.