# What We Know about Testing Embedded Software

**Vahid Garousi**, Wageningen University

**Michael Felderer**, University of Innsbruck

**Çağrı Murat Karapıçak**, KUASOFT A.Ş.

**Uğur Yılmaz**, ASELSAN A.Ş.

*// This article reports on a systematic literature review of 272 papers about embedded-software testing. The review provides an index to the body of knowledge in this area and could help practitioners choose suitable methods for their embedded-software testing projects. //*

**TO COST-EFFECTIVELY TEST** embedded software, practitioners and researchers have proposed various techniques, approaches, tools, and frameworks. However, obtaining an overview of the state of the art and state of the practice in this area can be challenging for practitioners or new researchers. The number of studies is simply too large. Consequently, by being unaware of the body of knowledge in this area, many companies waste much effort designing test approaches that they think are new but that already exist. Knowing that they can adapt or customize an existing test technique to their own context could lead to them saving much time and money.

In addition, on the basis of our experience and the opinions of other practitioners and researchers,[1,2] we believe that most practitioners don't actively read scientific papers, even though there are many papers in the area of embedded-software testing.

So, the need exists for review papers that summarize the area and that can serve as an index to the vast body of knowledge in this area. Such papers can give practitioners a snapshot of the knowledge out there without them having to find and read through each of the more than 200 papers in this area.

Although there have been state-of-the-practice papers on embedded-software engineering[3] and at least one review paper on embedded-software testing,[4] no paper has holistically studied the entire state of the art and state of the practice of embedded-software testing. Such an overview is essential for this area, which is driven equally by academia and industry.

To address this need, we conducted a systematic classification of the technical papers written by practitioners and researchers; this article summarizes the results. Previous reviews like this one have appeared on other topics—for example, agile development[5] and developer motivation[6]—and have proven useful in providing concise overviews of a given area.

## The Review Procedure

Our review and mapping employed the standard process for systematic literature reviews (SLRs)[7] and systematic literature mapping (SLM) studies[8] in software engineering. We performed the searches in the Google Scholar database. All four of us conducted all steps as a team. Our search string was "(test OR testing OR validation OR verification)


0740-7459/18/$33.00 © 2018 IEEE

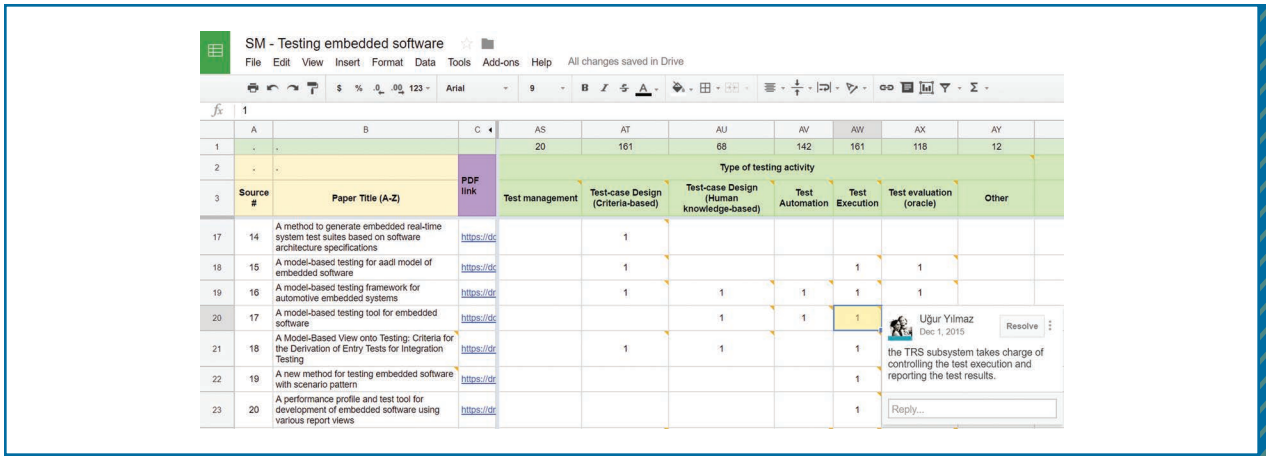


**FIGURE 1.** A screenshot of the online repository of the papers on embedded-software testing that we studied. You can access the repository at http://goo.gl/MhtbLD.

AND (embedded system OR embedded software)."

We initially compiled a pool of 560 candidate technical papers published in conference proceedings and journals. Then, we voted on each paper on the basis of a set of paper inclusion criteria and narrowed the pool to 272 papers.

The review addressed three main questions. First, what types of testing activities have been conducted and proposed? Inspired by books such as *Introduction to Software Testing*[9] and our SLR study[10] and SLM study,[11] we categorized testing activities as

- test management,
- criteria-based test-case design,
- human-knowledge-based test-case design,
- test automation,
- test execution,
- test evaluation, or
- other test activities (such as regression testing and test minimization).

Second, what types of artifacts do the proposed test techniques generate? After reviewing a large subset of papers and using iterative refinement, we categorized them as

- test-case requirements (not test inputs, but criteria that can be used to derive test inputs),
- test-case input,
- expected outputs,
- test code, or
- other test artifacts (such as test patterns and test documentation).

Finally, on what industries did the studies focus? Although some test techniques are generic—that is, they're applicable to all types of embedded software—some techniques are domain-specific. We categorized the industries as

- automotive;
- industrial automation and control;
- aviation, avionics, and space;
- mobile and telecommunications;
- home appliances and entertainment;
- medical;
- defense;
- railway;
- generic; or
- other.

The final pool and the online mapping repository are at http://goo.gl/MhtbLD; Figure 1 shows a screenshot of the repository. Throughout the rest of this article, we indicate specific papers using "P" and an ID number (for example, P34), using the IDs we used to label the papers in the online repository. For a more detailed description of our SLM process, see the document at http://goo.gl/by10jc. That document also discusses how we identified and addressed the potential threats to our review's validity and results.

## Academia and Industry Involvement

Figure 2 shows the number of studies of embedded-software testing published solely by academic researchers (those at universities and research centers or institutes), solely by practitioners (including those at corporate research centers), or as collaborative work, from the inception of this area until 2014.

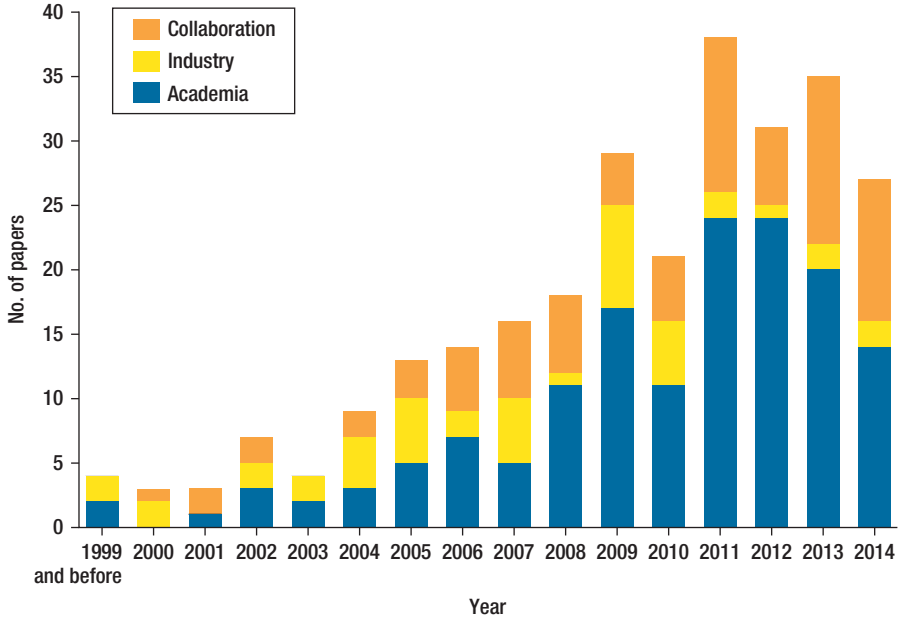


**FIGURE 2.** The number of studies of embedded-software testing from the inception of this area until 2014. Interest in this topic has risen steadily since the early 2000s.


**FIGURE 3.** A word cloud of the terms from all the paper titles, indicating the topics' popularity. Among the most popular topics were model-based and automated or automatic testing, test-case generation, and automotive systems.

The earliest paper was published in 1984.[12] As the figure shows, interest in this topic has risen steadily since the early 2000s for both researchers and practitioners. Because our study started in summer 2015, we decided to include the papers published until the end of 2014. The peak year in terms of the number of papers was 2011 (38 papers).

Regarding the type of authors of the papers in our pool, 151 papers were solely by academic researchers, 43 were by practitioners only, and 78 were collaborative. The companies that were the most active in conducting research and publishing papers in this area were Daimler (14 papers), Samsung (8 papers), and Nokia and Berner & Mattner (4 papers each).

## The Topics Studied

Figure 3 shows a word cloud of the terms in the paper titles, denoting the popularity of the topics covered. (We used the online tool at http://www.wordle.net.) Among the most popular topics were model-based and automated or automatic testing, test-case generation, and automotive systems.

## The Types of Testing Activities

Figure 4 shows a generic test process. Similarly to the testing of other types of software systems, embedded-software testing usually starts with test-case design (either criteria-based or human-knowledge-based). Using the derived test cases, test execution of the system under test (SUT) occurs. Finally, in test evaluation (using test oracles), the testing results are evaluated (pass or fail), and test verdicts are made.

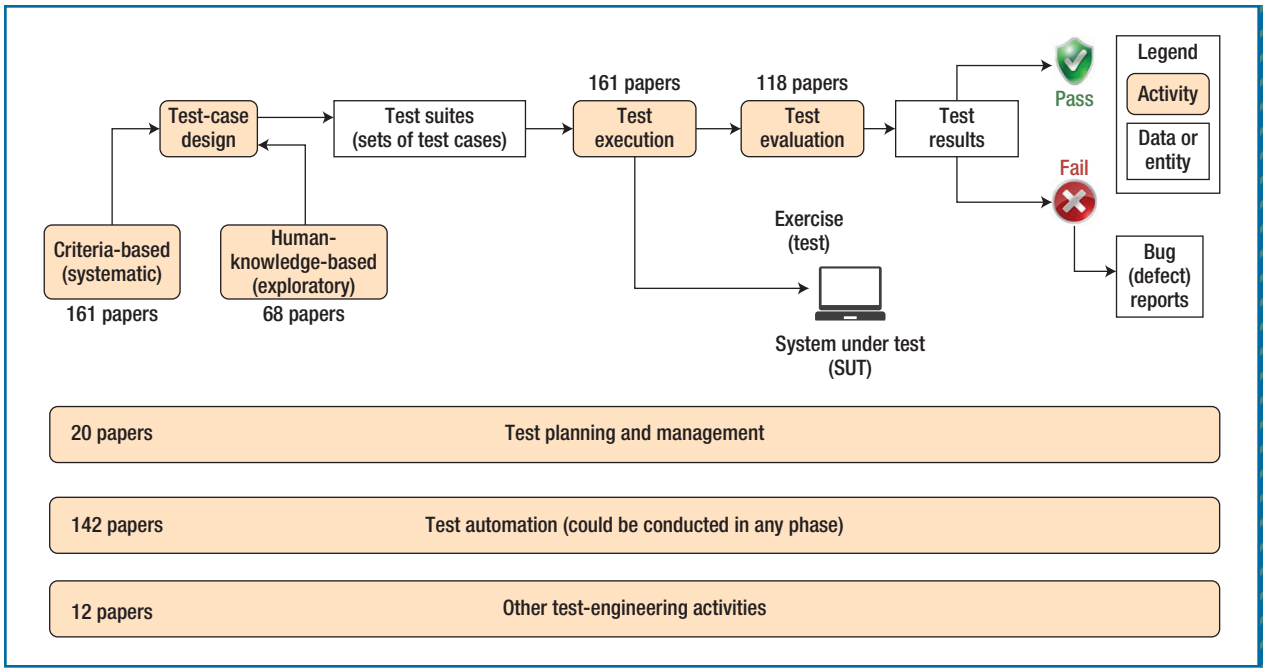


**FIGURE 4.** A generic test process showing the testing activities, along with the number of papers in our study that covered these activities. The sum of the numbers is higher than the number of papers because many papers discussed multiple activities.

Figure 4 also shows three cross-cutting activities: test management (which includes planning, control, and monitoring), test automation (which could occur in any phase), and other activities (for example, regression testing and test prioritization). Many people think of test automation as only for automated execution of test cases, but it has been successfully implemented in other testing activities—for example, test-case design and test evaluation.[13]

In addition, Figure 4 shows for each activity the number of papers in our study that discussed it. For example, 161 papers proposed techniques for criteria-based test-case design. As Figure 4 shows, there's a good mix of papers proposing techniques and tools for each test activity. The most frequently addressed testing activities in the papers were test execution, test automation, and criteria-based test-case design (for example, based on code coverage).

Next, we discuss some example studies for each test activity.

Two-hundred twenty-nine papers covered test-case design. For example, "Adaptation of State/Transition-Based Methods for Embedded System Testing" [P34] presented a test-case-design approach for generating test sequences from extended finite-state machines (FSMs). "Automated Generation of Test Cases from Output Domain and Critical Regions of Embedded Systems Using Genetic Algorithms" [P57] discussed how the researchers used genetic algorithms from partitioned input spaces to automatically generate test cases. They applied this approach to a system that monitored and controlled a nuclear reactor's temperature.

One-hundred sixty-one papers covered test execution. Because test execution is the main phase (activity) of testing, test-case design and the other activities in Figure 4 actually support test execution. That is, once test cases are designed, test engineers can execute them.

One-hundred eighteen papers covered test evaluation. For example, "An Approach for Test Derivation from System Architecture Models Applied to Embedded Systems" [P39] described how a test evaluation algorithm correlated the architectural information of the SUT (a mirror control system) with the functional requirements and verified that a certain side effect couldn't occur. The test suites in P39 programmatically validated that the SUT didn't move the mirror vertically if the mirror-position setting button was pulled to the right. In "Effective Test-Driven Development for Embedded Software" [P107],

the authors explained how they used a popular embedded unit-testing framework (in C) to write test scripts, including test evaluation using assertions—for example, `TEST_ASSERT_EQUAL_INT(7, Model_FeedbackVoltage)`. The SUT in that study was the speed control unit of vehicles.

One-hundred forty-two papers covered test automation. For example, "Teaching Cross-Platform Design and Testing Methods for Embedded Systems Using DICE" [P214] discussed a unit-testing framework called DICE (the DSPCAD Integrative Command Line Environment) and reported its use in teaching testing. "Model-Based Testing of Embedded Automotive Software Using MTest" [P155] proposed a model-based-testing toolset called MTest.

Twenty papers addressed test management. One such paper [P31] proposed a test process improvement model for embedded software development. In "Taming the Embedded Tiger: Agile Test Techniques for Embedded Software" [P212], the authors argued that, even with evolving hardware in the picture, agile methods work well provided that testers use the proper test strategies. For that purpose, the authors proposed an experience-based test strategy.

Twelve papers covered other test-engineering activities. For instance, "Integrating Test Levels for Embedded Systems" [P138] covered the reuse of test-case specifications throughout testing. "Rapid Embedded System Testing Using Verification Patterns" [P180] proposed a set of verification patterns (conceptually similar to design patterns) that lets software engineers develop automated test scripts effectively and efficiently.

The sum of the numbers in Figure 4 is higher than the number of papers because many papers discussed multiple testing activities. For example, "Improving the Accuracy of Automated GUI Testing for Embedded Systems" [P135] covered human-knowledge-based test-case design, test automation, test execution, and test evaluation.

To learn about the existing advances and potentially adopt or customize them, practitioners should review the studies listed at http://goo.gl/MhtbLD as categorized by test activity. For example, if a test team intends to conduct test-case design, it should review the 161 papers in the "criteria-based" category or the 68 papers in the "human-knowledge-based" category.

## The Types of Test Artifacts

Researchers have proposed various techniques to generate different types of test artifacts. Techniques for deriving test-case inputs (values) received the most attention (131 papers, or 48.1 percent). For example, the authors of "Applying Model-Based Testing in the Telecommunication Domain" [P52] applied model-based coverage criteria to derive test cases. They transformed UML models of the SUT to models in the Qtronic Modeling Language, a domain-specific language. Then, they used the Conformiq Qtronic tool to derive test cases. The authors applied their approach to the mobility management feature of a mobile-services switching-center server.

Another paper in this category is "Model-Based Testing of Embedded Automotive Software Using MTest" [P155], a joint work from two major German industrial firms. MTest uses classification trees to model test requirements. The researchers developed it to specify requirements and then used it to derive test sequences. They evaluated their approach on a real automotive system at Daimler Chrysler—the lane-change feature of a self-driving car.

Ninety-three papers addressed automated test-code generation (for example, in the xUnit test frameworks). For instance, "A Model-Based Testing Framework for Automotive Embedded Systems" [P16] reported an approach to generate test scripts in Python based on a specific form of abstract test cases. The authors applied the approach to an antilock-braking-system implementation of Volvo's brake-by-wire system prototype; it showed encouraging results.

Eighty-five papers presented approaches for generating test-case requirements. Test requirements usually aren't actual test input values; they're the conditions that can be used to generate test inputs. For example, "Automatic Test Case Generation and Test Suite Reduction for Closed-Loop Controller Software" [P72] discussed automatic test-case generation and test-suite minimization based on path coverage. The authors applied their approach to five controller programs based on implementations of medical protocols used at the Hospital of the University of Pennsylvania to determine the amount of insulin to pump.

Seventy-three papers dealt with the expected outputs (using test oracles). For example, "A Model-Based Testing for AADL Model of Embedded Software" [P15] proposed an approach for generating the expected outputs using specifications based on the Architecture Analysis and Design Language. The authors applied this approach to an experimental smart-curtain control system.

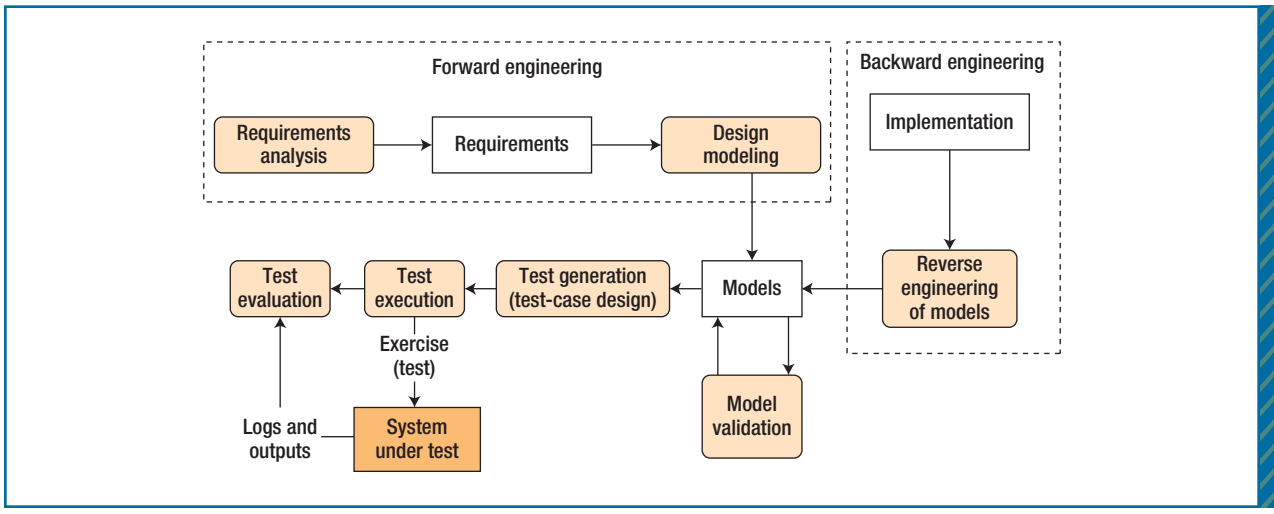


**FIGURE 5.** The general process of model-based testing, which was the most popular topic in the studied papers.

Nine other papers proposed other types of test artifacts—for example, test patterns [P83] and test documentation [P163].

## Model-Based Testing

The most popular technique for deriving test artifacts was requirements-based testing; 159 papers (58.4 percent) discussed it. Most of these papers (142, 52.2 percent) used model-based testing.

Figure 5 shows the general process of model-based testing, which employs forward engineering or backward engineering to develop models. Once models are validated, they can be used for test-case design. For example, FSMs and their extensions are frequently used to derive test-case sequences, employing coverage criteria such as all-transitions coverage. Given the wealth of knowledge and industrial evidence in model-based testing of embedded systems, companies that plan to implement systematic testing should review and consider this body of knowledge to potentially adopt some of the existing techniques.

In 46 papers, the authors used FSMs and their extensions (for example, timed FSMs) for model-based testing; examples include P16, P18, and P21. Twenty-eight papers utilized Matlab Simulink models; examples include P28, P29, and P54. Seventy-five papers discussed other types of models—for example, the Method Definition Language [P170] and UML sequence diagrams [P173].

Model-based testing fits in the scope of a larger development concept for embedded systems—that is, X-in-the-loop development, simulation, and testing [P160, P161, P194, and P208]. Subcategories in this area include model-in-the-loop, software-in-the-loop, processor-in-the-loop, hardware-in-the-loop, and system-in-the-loop. Such testing has gained wide acceptance in recent years owing to the increased adoption of model-based development in industry, especially in the automotive domain.[14]

## The Types of Industries

Figure 6 depicts our breakdown of the industries. The automotive sector received by far the most coverage (81 papers). Industrial automation and control came in second (33 papers). Aviation, avionics, and space came in third. Domains in the "Other" category included banking, e-government, and fire safety.

## The Benefits of This Review

Thanks to this study, we're assessing several existing model-based techniques for possible adoption or extension in our industry–academia collaborative projects. We've also observed that studies such as this one help bridge the industry–academia gap in software testing and software engineering in general.[15]

To further assess this study's benefits, we had several test engineers in the Turkish embedded-software industry review both an earlier version of this article and our online repository. Their general opinion was that such a review is an invaluable resource and can serve as an index to the body of knowledge in this area. Here's what one of the test engineers told us:

> *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 SUTs from our customers because of security*

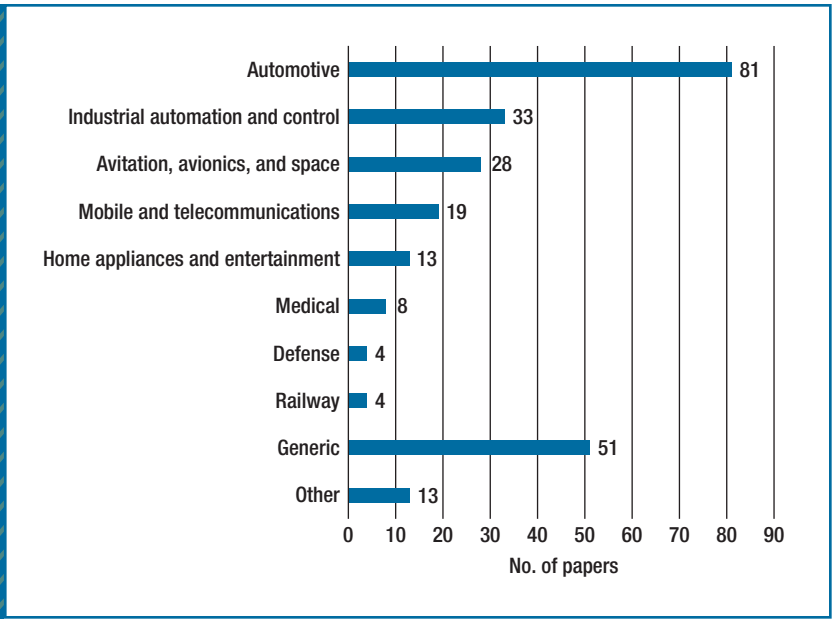


**FIGURE 6.** The industries (domains) represented in the papers. Domains in the "Other" category included banking, e-government, and fire safety.

*concerns. After reviewing this study, I wonder why we do not have any model of these SUTs. If we can create models of these SUTs (for example, model-in-the-loop), 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 testing activities. There are a lot of studies in the pool of this review studies that would benefit us. I think this idea is a major benefit for companies like ours. And I hope we may realize this idea [in the near future]. And as you have said in the paper, I believe that many companies are exactly reinventing the wheel."*

Although interest and progress in embedded-software testing have been considerable, more than half of the available papers only proposed a solution or reported experiences. They didn't provide empirical evidence of the presented approaches' effectiveness and efficiency. So, the need exists for more rigorous empirical studies providing industrial evidence on the effectiveness (for example, measured in terms of the defect detection rate) and efficiency of embedded-software-testing approaches in specific contexts (the extent to which a given test technique reduces the testing cost or effort). This will provide additional support for the selection of approaches.

We invite researchers to work on the open technical issues and needed solutions in this area. We also invite practitioners to

- exploit the vast body of knowledge in the many studies in embedded-software testing;
- assess the effectiveness of approaches, tools, and techniques;
- collaborate with researchers on the open issues; and
- report back in the form of papers about their experiences and lessons learned.

Finally, we're aware that new research and new results in this area will be regularly published. So, it will be important to regularly maintain and update the pool of studies. Because our dataset and pool are open access and publicly available, we invite researchers and practitioners to maintain or update the pool when another review study on this subject becomes necessary (in a few years).

## References

1. A. Tang and R. Kazman, "On the Worthiness of Software Engineering Research"; http://shidler.hawaii.edu/sites/shidler.hawaii.edu/files/users/kazman/se_research_worthiness.pdf.
2. D. Lo, N. Nagappan, and T. Zimmermann, "How Practitioners Perceive the Relevance of Software Engineering Research," *Proc. 10th Joint Meeting Foundations of Software Eng.* (ESEC/FSE 15), 2015, pp. 415–425.
3. C. Ebert and C. Jones, "Embedded Software: Facts, Figures, and Future," *Computer*, vol. 42, no. 4, 2009, pp. 42–52.
4. A. Banerjee, S. Chattopadhyay, and A. Roychoudhury, "On Testing Embedded Software," *Advances in Computers*, vol. 101, 2016, pp. 121–153.
5. T. Dybå and T. Dingsøyr, "What Do We Know about Agile Software Development?," *IEEE Software*, vol. 26, no. 5, 2009, pp. 6–9.
6. T. Hall et al., "What Do We Know about Developer Motivation?," *IEEE Software*, vol. 25, no. 4, 2008, pp. 92–94.
7. B. Kitchenham and S. Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*, ver. 2.3, tech. report

EBSE-2007-01, Keele Univ. and Univ. of Durham, 2007; http://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf.
8. K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for Conducting Systematic Mapping Studies in Software Engineering: An Update," *Information and Software Technology*, vol. 64, 2015, pp. 1–18.
9. P. Ammann and J. Offutt, *Introduction to Software Testing*, Cambridge Univ. Press, 2008.
10. S. Doğan, A. Betin-Can, and V. Garousi, "Web Application Testing: A Systematic Literature Review," *J. Systems and Software*, May 2014, pp. 174–201.
11. V. Garousi, Y. Amannejad, and A. Betin-Can, "Software Test-Code Engineering: A Systematic Mapping," *J. Information and Software Technology*, vol. 58, 2015, pp. 123–147.
12. M.A.J. Burford and F. Belli, "CADAS: A Tool for Designing Reliable Embedded Software and Supporting Testing 'in the Large,'" *Fehlertolerierende Rechensysteme* [Fault-Tolerant Computing Systems], vol. 84, Springer, 1984, pp. 101–112.
13. V. Garousi and F. Elberzhager, "Test Automation: Not Just for Test Execution," *IEEE Software*, vol. 34, no. 2, 2017, pp. 90–96.
14. V. Reyes, "Virtual Hardware in the Loop (vHIL): Earlier and Better Testing for Automotive Applications," 4 Feb. 2014; http://johndayautomotivelectronics.com/virtual-hardware-in-the-loop-earlier-and-better-testing-for-automotive-applications.
15. V. Garousi and M. Felderer, "Worlds Apart: Industrial and Academic Focus Areas in Software Testing," *IEEE Software*, vol. 34, no. 5, 2017, pp. 38–45.

## ABOUT THE AUTHORS

**VAHID GAROUSI** is an associate professor of software engineering at Wageningen University. His research interests include software engineering, software testing, and action research. He regularly collaborates with the software industry to improve software engineering in practice. Previously, he was an associate professor of software engineering at Hacettepe University, where most of the research reported in this article was conducted. Garousi received a PhD in software engineering from Carleton University. He has been an IEEE Computer Society Distinguished Visitor. Contact him at vahid.garousi@wur.nl.

**MICHAEL FELDERER** is a professor at the University of Innsbruck's Institute of Computer Science, a senior researcher at the Blekinge Institute of Technology, and the managing director of QE LaB Business Services. His research interests include software and security testing, software processes, requirements engineering, empirical software engineering, data analytics, and improving industry–academia collaboration. He also transfers his research results into practice as a consultant and speaker at industrial conferences. Felderer received a habilitation and PhD in computer science from the University of Innsbruck. Contact him at michael.felderer@uibk.ac.at.

**ÇAĞRI MURAT KARAPIÇAK** is a software engineer at KUASOFT A.Ş. and a PhD student in software engineering at Middle East Technical University. His research interests include software engineering and software testing. He received a BSc in electrical and electronics engineering from Hacettepe University. Contact him at cmkarapicak@kuasoft.com.

**UĞUR YILMAZ** is a test engineer at ASELSAN A.Ş. and an MSc student in software engineering at Hacettepe University. His research interests include software engineering and software testing. He received a BSc in electrical and electronics engineering from Middle East Technical University. Contact him at uguryilmaz@aselsan.com.tr.