Information and Software Technology 76 (2016) 92–117

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

When and what to automate in software testing? A multi-vocal literature review

Vahid Garousi[a,b,*], Mika V. Mäntylä[c]

[a] Software Engineering Research Group, Department of Computer Engineering, Hacettepe University, Ankara, Turkey
[b] Maral Software Engineering Consulting Corporation, Calgary, Canada
[c] M3S, Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, Finland

ARTICLE INFO

Article history:
Received 20 October 2015
Revised 25 April 2016
Accepted 28 April 2016
Available online 30 April 2016

Keywords:
Software test automation
Decision support
When to automate
What to automate
Multivocal literature review
Systematic literature review
Systematic Mapping study

ABSTRACT

Context: Many organizations see software test automation as a solution to decrease testing costs and to reduce cycle time in software development. However, establishment of automated testing may fail if test automation is not applied in the right time, right context and with the appropriate approach.
Objective: The decisions on when and what to automate is important since wrong decisions can lead to disappointments and major wrong expenditures (resources and efforts). To support decision making on when and what to automate, researchers and practitioners have proposed various guidelines, heuristics and factors since the early days of test automation technologies. As the number of such sources has increased, it is important to systematically categorize the current state-of-the-art and -practice, and to provide a synthesized overview.
Method: To achieve the above objective, we have performed a Multivocal Literature Review (MLR) study on when and what to automate in software testing. A MLR is a form of a Systematic Literature Review (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). We searched the academic literature using the Google Scholar and the grey literature using the regular Google search engine.
Results: Our MLR and its results are based on 78 sources, 52 of which were grey literature and 26 were formally published sources. We used the qualitative analysis (coding) to classify the factors affecting the when- and what-to-automate questions to five groups: (1) Software Under Test (SUT)-related factors, (2) test-related factors, (3) test-tool-related factors, (4) human and organizational factors, and (5) cross-cutting and other factors. The most frequent individual factors were: need for regression testing (44 sources), economic factors (43), and maturity of SUT (39).
Conclusion: We show that current decision-support in software test automation provides reasonable advice for industry, and as a practical outcome of this research we have summarized it as a checklist that can be used by practitioners. However, we recommend developing systematic empirically-validated decision-support approaches as the existing advice is often unsystematic and based on weak empirical evidence.

1. Introduction

According to a 2014 industrial survey of 1543 executives from 25 countries, testing and quality assurance of software-intensive systems accounts for roughly 26% of IT budgets [1], but lack of testing is even more costly. A 2013 study by the Cambridge University [2] states that the global cost of locating and removing bugs from software has risen to $312 billion annually and it makes up half of the development time of the average project.

Testing work can be roughly divided into automated and manual testing. In manual testing, a human tester takes the role of an end user and executes the features of given software under test (SUT) to ensure its behavior is as expected. To ensure completeness of testing, the tester often follows a written test plan that contains a set of test cases. Automated software testing means the automation of software testing activities [3]. In more specific terms, "*test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes with predicted outcomes*" [4]. The trade-off between automated and manual testing depends on various factors. Usually the first instinct of adopting test automation is just to apply it to do whatever the human testers were previously doing manually.

* Corresponding author.
E-mail addresses: vahid.garousi@hacettepe.edu.tr, vgarousi@gmail.com (V. Garousi), mika.mantyla@oulu.fi (M.V. Mäntylä).

http://dx.doi.org/10.1016/j.infsof.2016.04.015
0950-5849/

However, automation cannot replace manual testing completely or eliminate personnel costs [5]. The establishment of automated testing may fail if the test automation is not applied in the right context with the appropriate approach [3].

When properly implemented, test automation can considerably decrease the cost of testing and increase the quality of the software. However, according to a recent survey called "World Quality Report 2014–2015" [1], conducted by a French firm called Sogeti, only 28% of test cases are currently automated while the managers wish to increase this number in the future. As test automation becomes more and more mainstream in the software industry, the decisions on when and what to automate become very important since wrong decisions in this context can lead to disappointments and major wrong expenditures (resources and efforts).

Although, in the ideal world, many people have a vision of full automated testing, only 6% of practitioners supported this view according to a 2012 survey [6]. Other sources such as [7] and our surveys of test practices in Canada [8] and Turkey [9] confirm that practitioners think that it is not practical (or possible) to automate all tests, due to budget and time constraints. Deciding when and what parts of the system under test to be tested in an automated manner is a widely-asked question [10]. As of this writing (March 2016), a Google search for the phrase ("when to automate" software testing) returns more than 17,500 results, e.g., there are lots of online forums, debates and experience sharing on the topic. Furthermore, importance of the topic is highlighted in a practitioner-oriented book whose title "*Just enough software test automation*" [11] reflects that whether to automate software testing is not always a yes-no question. Other sources such as [12] make similar propositions: "*Like all testing activities, behind the decision to automate some tests is a cost and benefit analysis. If you get the analysis wrong, you'll allocate your sources inappropriately*". Various factors have been discussed in both the grey literature (e.g., blog posts and white papers) and the published (formal) literature (e.g., journal and conference papers) which would impact the trade-off between automated and manual testing [3].

Many researchers and practitioners have tackled the when- and what-to-automate questions in technical papers, online blogs and forums. However, no secondary studies (i.e., 'review' or survey papers) have been published on this topic, to review, gather and synthesize the knowledge on the above two W's of automated testing (what and when). To cover this gap between academic literature and industry's desire for practical knowledge, we performed a Multivocal Literature Review (MLR) study that covers not only academics studies and books, but also a wide area of the grey (unpublished non-research) literature available online, i.e., blog posts, white papers, presentation videos and tools. A MLR is a form of a Systematic Literature Review (SLR) which includes sources from both the grey and the published (formal) literature. Thus, in summary, the needs (motivations) for our study is to present a single source to researchers and practitioners, summarizing all the factors, approaches, guidelines and experience-based opinions w.r.t. the two W's of automated software testing (what and when), a need that many practitioners have personally expressed to us in our industry-academia interactions, e.g., [13–16].

The remainder of this paper is organized as follows. Section 2 presents the background and reviews the related work. Section 3 describes the goal, research method and review questions tackled in this study. Section 4 discusses the paper and source selection process. Section 5 presents the systematic map (i.e., classification scheme) which was iteratively developed in our study. Section 6 presents the results of the MLR. Section 7 summarizes the results and the implications of the MLR for researchers and practitioners, and discusses limitations of our study and its results. Finally, Section 8 concludes this study and states the future work directions. The reference section at the end of the study is divided into two parts: pool of the sources reviewed in the MLR is listed first and then the other references used in this study.

2. Background and related work

In this section, we briefly provide first a review of test automation and the 'when-' and 'what-to automate' issues. We then present a brief background on multivocal literature reviews (MLRs) since it is a relatively new terminology in software engineering. We then review the related work.

2.1. Review of test automation and the 'when-' and 'what-to automate' issues

Automated software testing means the automation of software testing activities usually conducted by humans. Test automation is conducted using software tools referred to as test tools [3]. Test automation has a history of over two and half decades, since around 1990 [17], and can lead to many benefits, e.g., cost savings and higher software quality [19].

Any major commercial or open-source software nowadays includes automated test suites to verify its functionality. This is specially the case for software projects which evolve through many versions since automated testing pays off the most in the case of regression and repetitive testing. For many large-scale systems, automated test suites are constantly increasing in size and complexity. For example, the code-base of version 2.1 of the Android OS had 357,933 LOC of JUnit test code which accounted for 17.1% of the Java code-base of the OS, i.e., 2090,904 LOC (data from [18]). One can easily imagine the major effort needed to develop such a test suite in the first place, ensure its integrity (quality) and to co-maintain it alongside the production code.

In a Google Tech Talk in November 2005 [19], Jeff Feldstein, then the Manager of Software Development in Cisco Systems, referred to the extent of test automation in the context of routers built by Cisco and mentioned that: "*We designed a test system that probably is as complicated as the system itself.*", referring to the amount of automated test suites developed for a particular router model.

As an example of automated test scripts, Fig. 1(a) shows the automated user interface (UI) test case developed using the Selenium tool (www.seleniumhq.org) for testing an issue tracking web application called JIRA. This test case verifies the enter-an-issue feature of JIRA. The four conceptual steps of testing (setup, exercise, verify, and teardown) have been explicitly highlighted in Fig. 1(a).

As another example of automated test scripts, Fig. 1(b) shows the unit test code for the Android platform in JUnit checking whether the emergency number (911) is properly set. In this test case, we can see that a good practice, the so-called *test pattern* [20], has been used in modularizing the test code by putting a set of specific verification rules under a function named `assertIsValidEmergencyCallerInfo()` and calling it from this example test method. According to many studies, it has been empirically observed that using test patterns increases the quality of test suites, e.g., [20].

To assess the popularity of test automation in the community, we conducted a search for software testing books in Google Books search engine (books.google.com) and compiled a list of all testing books and also only those focusing on test automation, accessible in an online spreadsheet https://goo.gl/rAbDbC. In order to keep the workload manageable, we only identified the test automation books published between 2010 and 2014. The trend is shown in Fig. 2. In total, 247 software testing books have been published in the period of 1979-2014. Between 2010 and 2014, 78 of the total 113 testing books (69%) have focused on test automation. This denotes the high popularity of test automation in the community.

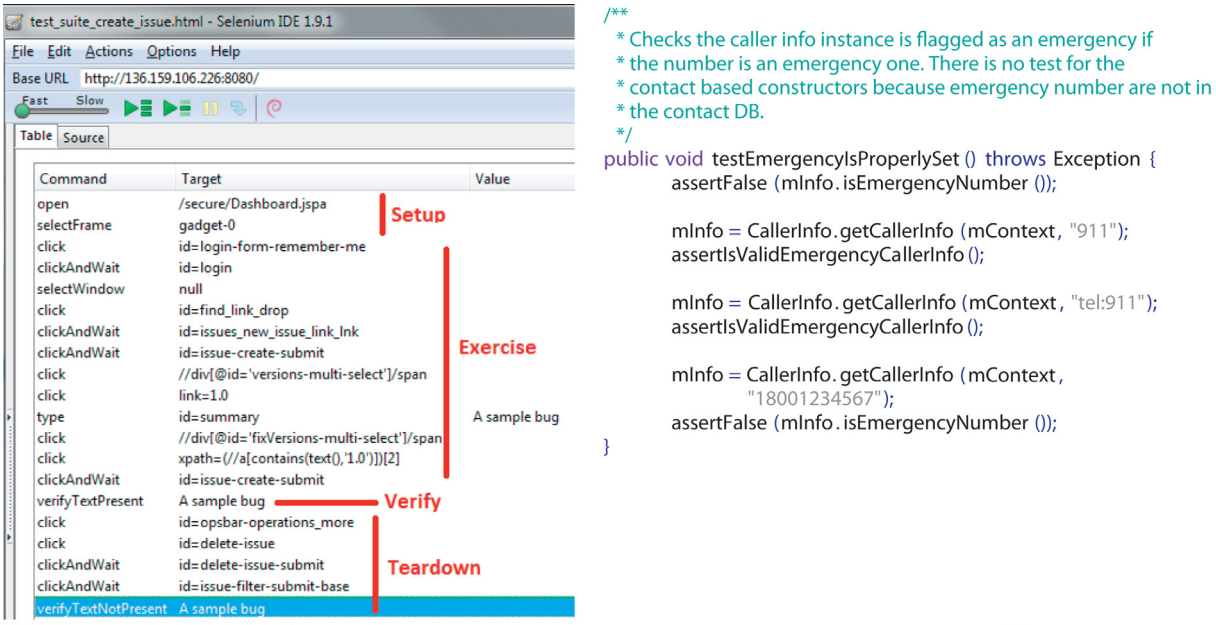


Fig. 1. Two example automated test-cases.

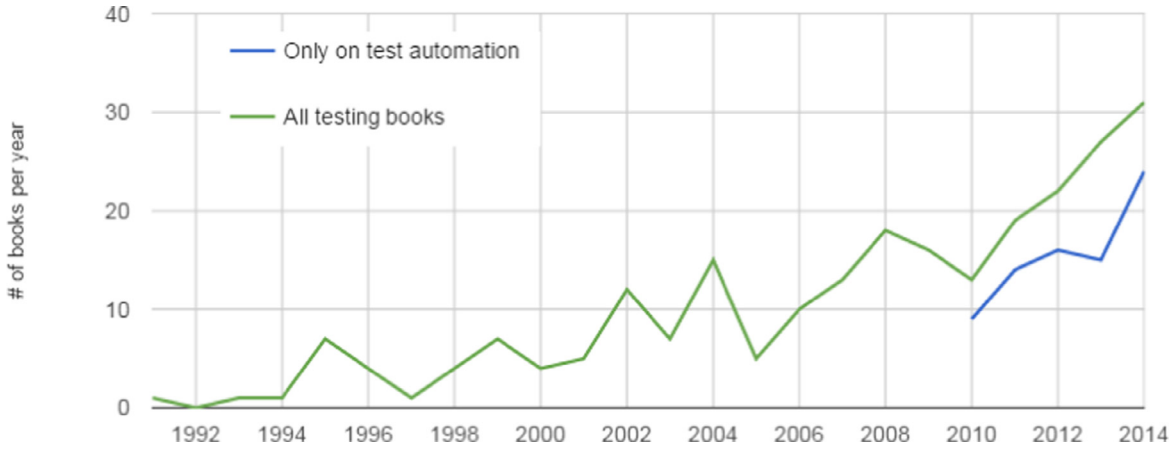


Fig. 2. Publication trends of all testing books and only those focusing on test automation.

While test automation mostly started with the automation of test execution, it has expanded to other areas of testing too, e.g., automated test-case design, automated test scripting, and automated defect reporting. Going beyond just automation of test execution, large companies such as Microsoft and Google have benefited a lot from automating different test activities, as reported in two recent books: '*How We Test Software at Microsoft*' (2008) [21] and '*How Google Tests Software*' (2012) [22].

Typically, a test process consists of several steps from test planning to test specification (test-case design), execution, and reporting [23], each of which can be either done manually or be automated. To better understand how automation is used during the test process (and not just during execution), based on many sources of test automation, we present below six testing activities where a large potential for automation could be seen:

1. Test-case design: designating a list of test cases or test requirements to satisfy coverage criteria, other engineering goals, or based on human expertise (e.g., exploratory testing).
2. Test scripting: documenting test cases in manual test scripts or automated test code
3. Test execution: running test cases on the software under test (SUT) and recording the results
4. Test evaluation: evaluating the results of testing (pass or fail), also known as test verdict
5. Test-result reporting: reporting test verdicts and defects to developers, e.g., via defect (bug) tracking systems
6. Test management and other test engineering activities: Test management includes activities such as planning, control, monitoring, and effort estimation. Other test activities include test suite minimization and regression test selection.

An overview of these steps is shown in Fig. 3, which depicts an overview of automation across the software testing process.

Many researchers and practitioners have proposed decision-support approaches for 'when-' and 'what-to automate', which this MLR aims to review, synthesize and present in a single source. We review three example sources [24–26] from that pool next.

When deciding what parts of the system should (not) be automated, several factors need to be considered. Practitioner testers from Microsoft recommended three major factors to be considered [24]: "(1) rate of change of what we are testing: the less stable, the more automation maintenance costs, (2) frequency of test

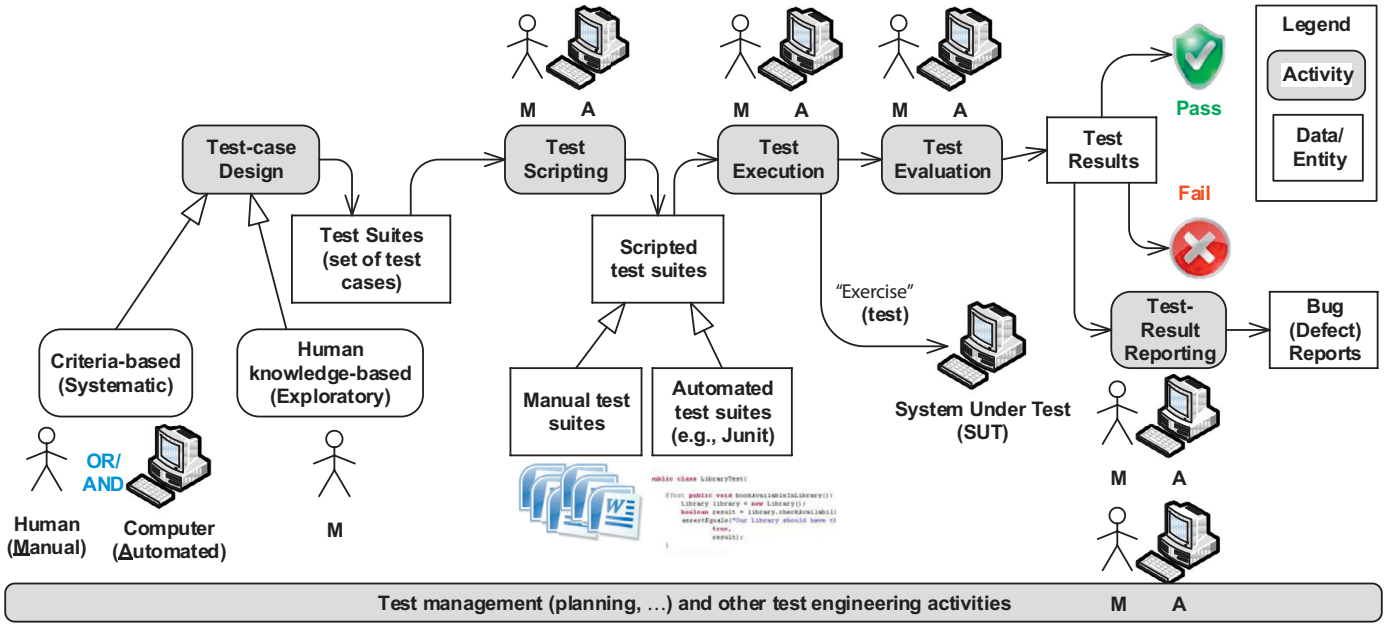


Fig. 3. An overview of automation across the software testing process.

execution: How important is each test result and how expensive is to get it?, and (3) usefulness of automation: Do automated tests have continuing value to either find bugs or to prove important aspects about your software, like scenarios?"

Inside a book on testing [25], generic tips to address the "what to automate" problem were provided. According to this book, certain types of testing such as stress, reliability, and regression testing are amenable types for being automated. Due to the repetitive nature of the regression testing, automation can save significant time and effort and the gained time can be effectively utilized for ad-hoc testing and other more creative avenues [25]. Similar to the challenge in our context, finding the right combination of test cases to be automated, [25] suggested that while starting automation, the effort should focus on areas where good scenarios in terms of ROI exist [25], however no systematic approach was provided.

A comprehensive checklist to support decision making for 'when-' and 'what-to automate' issues was provided by another book on test automation [26], which has considered almost all the above mentioned factors. Number of executions, covering critical paths, error-prone areas, data-driven features, number of supported hardware and software, and also having promising ROI are the main factors discussed in this book [26].

2.2. Multivocal literature reviews

Systematic Literature Review (SLR) and Systematic Mapping (SM) studies have become quite popular in software engineering. While SLR and SM studies are quite valuable, researchers have reported that "*the results of a SLR or a SM study could provide an established body of knowledge, focusing only on research contributions*" [27]. Since those studies do not include the "grey" literature (non-published, nor peer-reviewed sources of information), produced constantly in a very large scale by practitioners, those studies do not provide much insight into the "state of the practice". For a practical (practitioner-oriented) field such as software engineering, synthesizing and combing both the state-of-the art and – practice is very important. Unfortunately, it is a reality that a large majority of software practitioners do not publish in academic forums [28], and this means that the voice of the practitioners is limited if we do not consider grey literature in addition to academic literature in review studies.

2.2.1. MLRs in other fields

SLRs which include both the academic (formal) and the grey literature were termed in early 1990's in other fields, e.g., education, as Multivocal Literature Reviews (MLR), e.g., [29]. The main difference between a MLR and a SLR or a SM is the fact that, while SLRs and SMs use as input only academic peer-reviewed articles, in MLRs, grey literature, such as blogs, white papers and web-pages, is also considered as input [27]. A multivocal synthesis is suggested as an appropriate tool for investigations in a field "*characterized by an abundance of diverse documents and a scarcity of systematic investigations*" [30]. Researchers also believe that: "*another potential use of multivocal literature reviews is in closing the gap between academic research and professional practice*" [29].

While the notions of "MLR" and "multivocal" have been used in the community, still many sources use the "grey" literature terminology and whether/how to include them in SLRs, e.g., [31–33]. For example, [31] discusses the advantages and challenges of including grey literature in state-of-the-evidence reviews, in the context of evidence-based nursing. [32] discusses the challenges and benefits of searching for grey literature in SLRs.

A 1991 paper [29] discussed rigor in MLRs and proposed a method based on exploratory case study to conduct MLRs rigorously. Hopewell *et al.* [34] conducted a review of five studies, in the area of evidence-based medicine, comparing the effect of the inclusion or exclusion of 'grey' literature in meta-analyses of randomized medical trials. The issue of the grey literature is such important that there is even an International Journal on the topic of Grey Literature (www.emeraldinsight.com/toc/ijgl/1/4).

2.2.2. MLRs in software engineering

The 'multivocal' terminology has only been recently started to appear in the SLRs in software engineering, i.e., since 2013 in [35]. We found only three SLRs in software engineering which explicitly used the 'multivocal' terminology: [27,35,36]. [27] is a 2015 MLR on the financial aspect of managing technical debt. [35] is a 2013 MLR on technical debt. [36] is a 2015 MLR on iOS applications testing.

Table 1
A selected list of 15 of the 102 of the secondary studies in software testing (the full list can be found in [40]).

Type of secondary study	Secondary study area	Year of publication	Reference
SMs	Search-based testing for non-functional system properties	2008	[41]
	Product lines testing	2011	[42]
	Graphical user interface (GUI) testing	2013	[43]
	Test-case prioritization	2013	[44]
	Software test-code engineering	2014	[45]
SLRs	Model-based testing	2007	[46]
	Automated acceptance testing	2008	[47]
	Mutation testing for Aspect-J programs	2013	[48]
	Web application testing	2014	[49]
	Testing scientific software	2014	[50]
Regular surveys	Testing finite state machines	1996	[51]
	Regression testing minimization, selection and prioritization: a survey	2012	[52]
	Testing in SOA	2013	[53]
	Test-case generation from UML behavioral models	2013	[54]
	Test oracles	2014	[55]

Many other SLRs have also included the grey literature in their reviews and have not used the 'multivocal' terminology, e.g., [37]. A 2012 MSc thesis entitled "*On the quality of grey literature and its use in information synthesis during systematic literature reviews*" [38] explored the state of including the grey literature in SLRs in software engineering. Two of the research questions (RQs) in that study were: (1) What is the extent of usage of grey literature in SLRs? and (2) How can we assess the quality of grey literature? The study found that the ration of grey evidence in the SLRs were only about 9%, and the grey literature included concentrated mostly in recent past (~48% between years 2007–2012).

A recent ongoing work [39], in which the authors are involved, aimed at raising the need for (more) MLRs in software engineering. We did so by raising and addressing two RQs: (1) What types of knowledge are missed when a SLR does not include the multivocal literature in a SE field? and (2) What do we, as a community, gain when we include the multivocal literature and conduct MLRs? To answer these RQs, we sampled several example SLRs and MLRs, and identified the missing and the gained knowledge due to excluding or including the multivocal literature.

2.3. Review of secondary studies in software testing

No secondary studies have been published on the when- and what-to-automate questions. Thus, as the related work, we briefly review the secondary studies in software testing. By a literature search, we were able to identify a large number (102) of secondary studies in software testing, which we have listed in an online spreadsheet [40]. Secondary studies are usually of three types: SM studies, and SLRs and regular surveys. As a snapshot, we show a randomly-selected list of 15 of the 102 secondary studies in software testing in Table 1, five in each of the above three categories.

No secondary study has been reported in the exact scope of when and what to automate in software testing. A somewhat related work is [6] which reported an SLR and a practitioner survey on benefits and limitations of automated software testing. In addition, another major difference between our work and [6] is that [6] did not include grey literature, while we do so in our work.

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.* [56] have discussed the educational value of SM in the software engineering literature for the students. Usefulness of SLRs for practitioners has been studied in a number of non-SE fields, such as in disability research [57], in education research [58], and in health and social care [59].

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. Goal and review questions

The goal of this study is to systematically map (classify), review and synthesize the state-of-the-art and –practice for answering the questions of when and what to automate in software testing, to find out the recent trends and directions in this field, to identify opportunities for future research, from the point of view of researchers and practitioners in this area. Based on the above goal, we raise the following review questions (RQs). Note that previous SM and SLR studies called these types of questions as "research" questions, but based on the established terminology in other fields, such as in education research [58], we refer to them as "review" questions instead.

- RQ 1-mapping of sources by contribution and research facets:
  - RQ 1.1- mapping by contribution facet: How many studies present methods, techniques, tools, models, metrics, or processes for the when/what to automate questions? Mapping of studies by contribution facet is a usual practice in many SM studies, e.g., [60–62], as proposed by Petersen et al. [63,64]. Answering this RQ will enable us to assess whether the community as a whole has had more focus on developing new techniques, or more focus on developing new tools, etc.
  - RQ 1.2-mapping by research facet: What type of research methods have been used in the studies in this area? Some studies only propose solutions without extensive validations, while some other studies present in-depth evaluation of their approach (e.g., rigorous empirical studies). Petersen et al. [63,64] has also proposed guidelines to classify the research approach of studies, which we will use to answer this RQ. The rationale behind this RQ is that knowing the breakdown of the research area with respect to (w.r.t.) research-facet types will provide us with the maturity of the field in using empirical approaches.
- RQ 2-factors considered for deciding when/what to automate (the core contribution of this study): What factors are considered in the when/what questions? This RQ is the main contribution of this study in particular for practitioners as this synthesizes all the factors discussed in the primary sources. We aim at grouping and calculating the frequency of such factors.
- RQ 3-tools: What tools have been proposed to support the when/what questions?

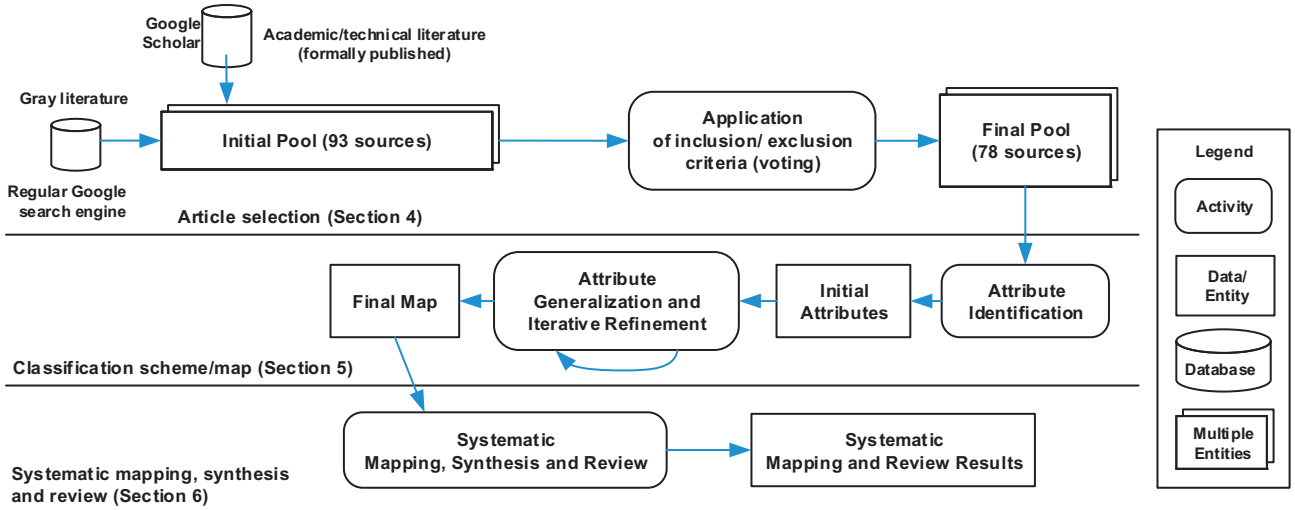


Fig. 4. An overview of the research process used to conduct this study.

- RQ 4-software systems under test or projects under study: What are attributes of those systems and projects?
  - RQ 4.1-software systems or projects under analysis: How many software systems or projects under analysis have been used in each source? One would expect that a given paper or article applies the proposed idea to at least one system to show its effectiveness.
  - RQ 4.2- domains and types of the software systems or projects under analysis: What are the domains of the software systems or projects under analysis that have been studied in the sources (e.g., embedded, safety-critical, and control software)? Also, what types of software systems have been studies in the sources (i.e., open-source, commercial, or academic experimental systems)?
  - RQ 4.3-types of measurements: What types of measurements, in the context of the software systems under analysis, to support the when/what questions have been provided? We wanted to know the ratio of sources which have conducted quantitative measurements to assess the proposed approaches.

3.2. Research method

As discussed above, this study is carried out based on the guidelines provided by [65,66]. In designing the methodology, methods from several other SM and SLR studies such as [67–69] were also incorporated. The process that lies at the basis of this study is outlined in Fig. 4, which consists of three phases:

- Article selection (Section 4).
- Development of the systematic map (Section 5).
- Systematic mapping, synthesis and review (Section 6).

The process starts with article selection from various academic sources. Then, a systematic map is systematically developed. The systematic map is then used to conduct systematic mapping and results are then synthesized and reported. Details of the above phases are described in Sections 4, 5, and 6.

4. Selection of formally-published papers and sources from the grey literature

Let us recall from our MLR process (Fig. 4) 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 which was conducted through a voting process between the two authors (Section 4.2).
- Finalizing the pool of articles and the online repository (Section 4.3).

4.1. Source selection and search keywords

This study followed the systematic guidelines for performing systematic literature review and mapping studies [63,64,66]. Using the guidelines and also based on our previous experience [6,43,70], we first defined the set of Review questions (RQs), the scope of the study, and the search strings.

We then performed the searches in the Google search engine (for the grey literature) and a popular academic search engine (i.e., Google Scholar) for the formally published papers, included/excluded articles, and finally extracted, analyzed and synthesize the factors from the relevant articles. The two authors conducted all the steps as a team. Our search strings were:

(a) "when to automate" testing.
(b) "what to automate" testing.
(c) decision automated software testing.
(d) decision software test automation.

Details of our source selection and search keywords approach were as follows. Both authors did independent searches with the search strings, and during this search, both authors already applied inclusion/exclusion criterion for including only those which explicitly addressed either of the two W's questions. As a result of the initial search phase, we ended up with a rather small initial pool of 93 articles, out of which 15 were later removed during the analysis and exclusion phases.

Typically in SM and SLR studies, a team of researcher includes all the search results in the initial pool and then separately performs the inclusion/exclusion as a separate step. This results in huge volumes of irrelevant papers. For example, in a SLR and practitioner survey in which the second author was involved [6], the team of researchers started with an initial pool of 24,706 articles but out of those only 25 were found relevant finally. This means high effort due to the very relaxed selection and filtering in the first phase. However, on the other hand, in two other SM studies in which the first author was involved, i.e. [43,70], the initial filtering was more rigorous and the reduction of the paper sets were

as follows: (1) from an initial pool of 230 papers to a final pool of 136 papers in [43], and (2) from an initial pool of 72 papers to a final pool of 60 papers in [70]. In those latter studies, the teams of researchers found the process to be more effective and efficient, while at the same time, the quality of the selection and results was not impacted. We thus followed the same approach used in [43,70] in this study as well.

We also utilized the relevance ranking of the search engines (e.g., Google's PageRank algorithm) to restrict the search space. For example, if one applies search string (c) above to the Google search engine, 1,330,000 results would show as of this writing (April 2015), but as per our observations, relevant results usually only appear in the first few pages. Thus, we checked the first 10 pages (i.e., somewhat a search "saturation" effect) and only continued further if needed, e.g., when the results in the 10th page still looked relevant.

Typically, in most SM and SLR studies [63,64,66], the searches are only targeted for papers in the academic literature [63,64]. Since a large majority of software practitioners do not publish in academic forums [28], this means that the voice of the practitioners is limited if we do not consider grey literature in addition to academic literature. We thus included blog-posts and other types of grey literature as well (e.g., white papers and even credible YouTube videos). Although, blogs do not usually qualify as scientific evidence, we see them as important outlets through which the voice of the practitioners can be studied. We believe that inclusion of the gray literature increases the relevance and usefulness of this study [71] to both researchers and practitioners.

Utilizing grey literature in other fields such as health sciences has been explored, but the field of SE seems to be quite marginal in this issue and not many SLR/SM studies or regular papers do not cite or get material from the SE grey literature.

4.2. Inclusion/exclusion criteria and their application

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:

- We included sources in the area of automated testing ROI calculations since they could be used as decision support mechanisms for balancing and deciding manual versus automated software testing
- Sources providing decision support for the two questions "what to automate" and "when to automate"

Sources which did not meet the above criteria were excluded. A few example excluded sources are [3,72–74]. The title of [3] is "*Trade-off between automated and manual software testing*" which looks much related in the first sight, but it addresses neither of the two W&W questions, thus it was excluded. Entitled "*Common mistakes in test automation*", [72] also did not target any of the two W&W questions. Included in the first candidate pool, [73] reported some empirical observations on software testing automation, but did not target any of the two W&W questions. As the last example, entitled "*Seven steps to test automation success*", while [74] presenting interesting recommendations about test automation, it also did not touch either of the two questions.

To ensure including all the relevant sources as much as possible, we conducted forward and backward *snowballing* [75], 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) [75].

4.3. Final pool of sources and the online repository

Our final pool of sources included a total of 78 sources, which had: 17 technical and scientific sources [Sources 1–17], 9 books and book chapters [Sources 18-26], 46 internet articles and white papers [Sources 27–72], 2 YouTube videos [Sources 73, 74], and 4 tools [Sources 75-78] assisting decision makers in the when/what questions. To ensure transparency our final repository can be found in http://goo.gl/zwY1sj.

We report next a summary on the trends in the final pool of sources, based on these aspects:

- Number of sources per year (growth of attention in this area) by literature type (formally published versus grey literature)
- Number of sources by source type
- Number of sources by type of contributors (contributions from academia versus industry)

4.3.1. Number of sources per year (growth of attention in this area)

Fig. 5 shows the cumulative number of sources per year by literature type (formally published versus grey literature). We can see that sources in both literature categories have been on a steady increasing trend. The grey literature in this area seems to have surpassed the formally published literature starting around year 2006, denoting the higher attention of practitioners on this important topic in test automation. An alternative explanation may be related to the popularity of Blogging that has been increasing in recent years as many testing professional are using it for communication and professional development. Also the fact that older grey literature might not be available anymore due to lack of systematic archiving of such works has probably affected the low amount of older grey literature. We can see from the figure that the topic is expected to get increased attention in the years to come as well.

4.3.2. Number of sources by source type

Fig. 6 depicts the number of sources by source type in each of the two categories: formally-published versus grey literature. We divided the formally-published and grey literature, respectively, into five and three categories as shown.

In the formally-published literature category, there were 9 conference papers, 9 books or book chapters, 5 journal papers, 2 workshop papers and 1 thesis. In the grey literature category, there were 46 online articles and white papers, 4 tools and 2 YouTube videos.

As discussed in Section 4, since a large majority of software practitioners do not publish in academic forums [28], this means that the voice of the practitioners is limited if we do not consider grey literature in addition to academic literature. This paper takes one step in that direction by including and reviewing the knowledge base and voice of software practitioners as portrayed by grey literature.

4.3.3. Number of sources by type of contributors (contributions from academia versus industry)

Fig. 7 shows the number of sources by type of contributors, in which there are three categories: (1) a paper having all academic authors, (2) all industry authors, and (3) a mix of academic and industry authors (collaborative work). The rationale behind this analysis is to assess how active academia and industry have each been in addressing the W&W questions. Mainly due to the high number of sources from the grey literature, sources written by industry authors are the majority. There were 6 sources by academic authors [Sources 2, 4, 6, 7, 11, 15], and only 3 collaborative works [Sources 1, 3, 17]. The latter denote the need for more industry-academic collaborations in this area.

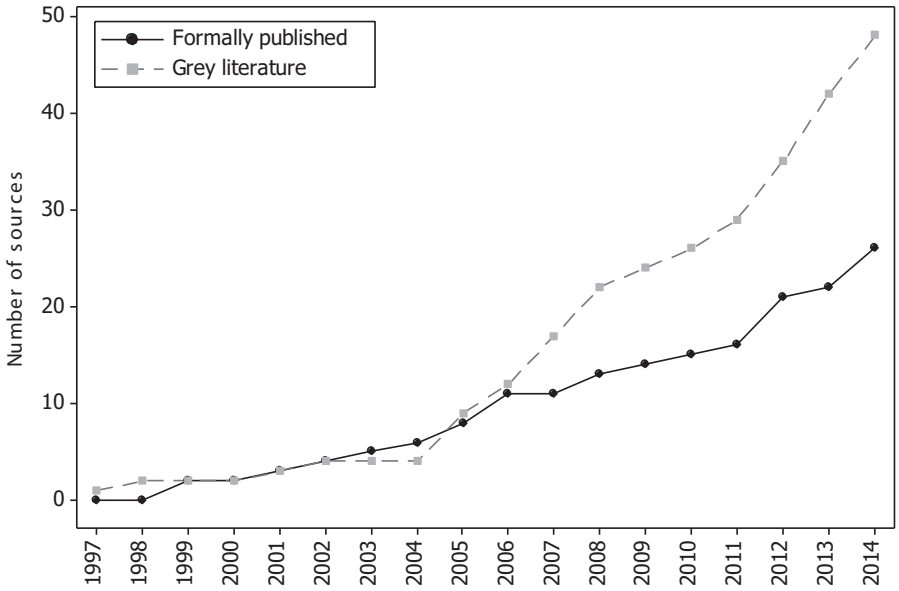


Fig. 5. Cumulative number of sources per year.

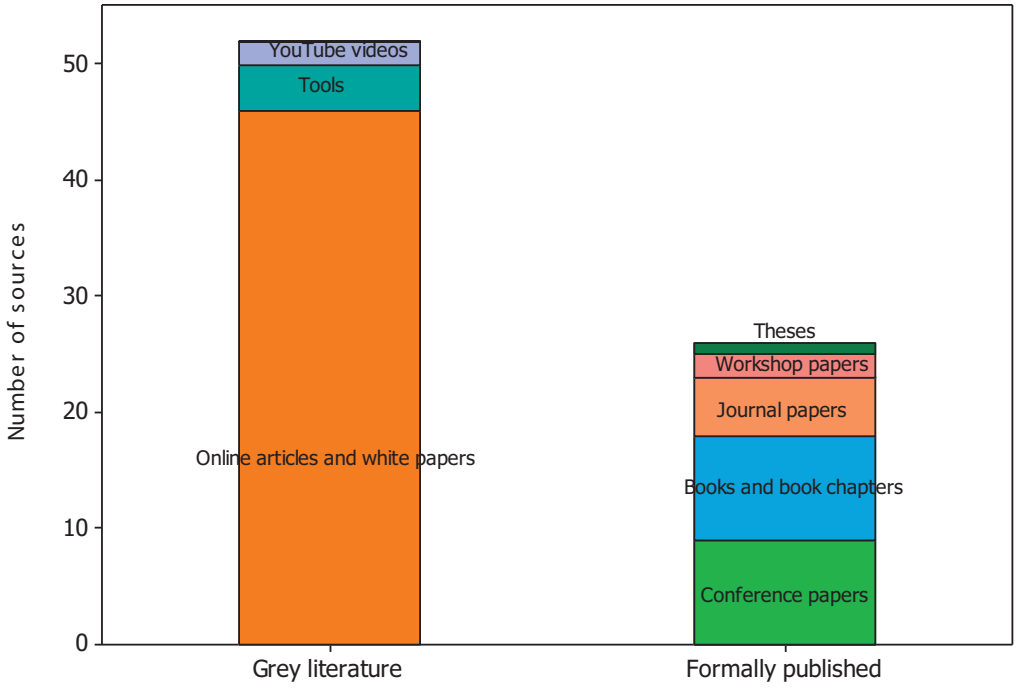


Fig. 6. Number of sources by type.

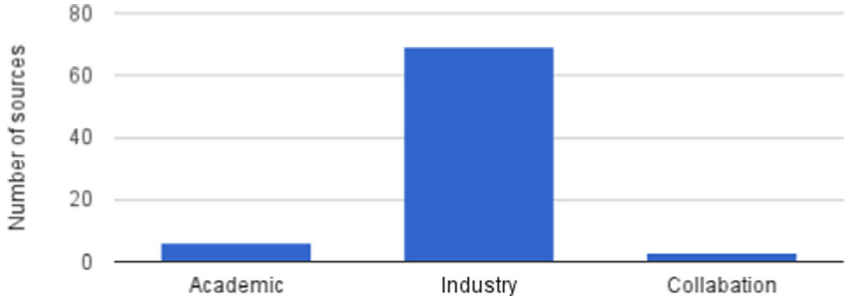


Fig. 7. Number of sources by type of contributors.

Table 2
Systematic map developed and used in our study.

RQ	Attribute/Aspect	Categories	(M)ultiple/ (S)ingle
1	Contribution type	Heuristics/guideline, method (technique), tool, metric, model, process, empirical results only, other	M
	Research type	Solution proposal, validation research (weak empirical study), evaluation research (strong empirical study), experience studies, philosophical studies, opinion studies, other	S
2	Factors considered for deciding when/what to automate	A list of pre-defined categories (Maturity of SUT, Stability of test cases, 'Cost, benefit, ROI', and Need for regression testing) and an 'other' category whose values were later qualitatively coded (by applying 'axial' 'open' coding)	M
3	Decision-support tools	Name and features	M
4	Attributes of the software systems under test (SUT)	Number of software systems: integer	M
		SUT names: array of strings	
		Domain, e.g., embedded systems	
		Type of system(s): Academic experimental or simple code examples, real open-source, commercial	
		Test automation cost/benefit measurements: numerical values	

5. Development of the systematic map and data-extraction plan

Iterative development of our systematic map is discussed in Section 5.1. Section 5.2 presents the final systematic map. Section 5.3 discusses our data-extraction and synthesis approach.

5.1. Systematic map

To develop our systematic map, as shown in Fig. 4, 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 (hosted in the online Google Docs spreadsheet [76]) to facilitate further analysis. Our next goal was to categorize the studies in order to begin building a complete picture of the research area. Though we did not a-priori develop a categorization scheme for this project, we were broadly interested in: (1) factors considered for the W&W questions, and (2) types of systems under test which were used in studies. We refined these broad interests into a systematic map using an iterative approach.

Table 2 shows the final classification scheme that we developed after applying the process described above. In the table, column 1 is the list of RQs, column 2 is the corresponding attribute/aspect. Column 3 is the set of all possible values for the attribute. Finally, column 4 indicates for an attribute whether multiple selections can be applied. For example, in RQ 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 (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.).

We believe most of the categories in Table 2 are self-explanatory, except for those for contribution and research types (RQ 1) which are explained in the next two sub-sections.

Petersen *et al.* [65] proposed the classification of contributions into: method/technique, tool, model, metric and process. These types were adapted in our context. We also added another type: survey or empirical results, since we found that many studies contribute such results. If a study could not be categorized into any above-mentioned types, it would be placed under "Other".

The second set of categories in our scheme deals with the nature of the research method used in each source. These categories were influenced by categories described by Petersen *et al.* [65] although they are not an exact replica. Our aim is to provide insights into the empirical foundation being developed by the body of research. The "research type" categories include:

- Solution proposal: A study in this category proposes a solution to a problem. The potential benefits and the applicability of the solution are shown only by a small example or a good line of argumentation.
- Validation research (weak empirical study): A study in this category provides preliminary empirical evidence for the proposed techniques or tools. Formal empirical methods (e.g., hypothesis testing, case studies, technical action research, controlled experiments) are not used or their use is not sufficient (e.g. experiments with low number of subjects or inadequately conducted case studies)
- Evaluation research (strong empirical study): These studies go further than studies of type "Validation research" by using strict and formal experimental methods (e.g., hypothesis testing, case studies, controlled experiment) in evaluating novel techniques or tools in practice or practice like settings. We essentially interpret Validation research (weak empirical study) and Evaluation research (strong empirical study) as a "rubric". By using the examples given (using formal empirical methods or not), we were able to classify a given empirical study under weak or strong empirical study.
- Experience studies: Experience studies explain how something has been done in practice, based on the personal experience of the author(s).
- Philosophical studies: These studies sketch a new way of looking at existing things by structuring the area in form of a taxonomy or conceptual framework.
- Opinion studies: These studies express the personal opinion of the author(s) around whether a certain technique is good or bad, or how things should been done. They do not significantly rely on related work or research methodologies.
- Other: A catchall category in the event that the work reported in a study does not fit into any of the above research types.

5.2. Data extraction and qualitative synthesis

To extract data, the studies in our pool were reviewed with the focus of each RQ. We also incorporated as much explicit 'traceability' links between our mapping and the primary studies as possible. Fig. 8 shows a snapshot of the online repository (spreadsheet hosted on Google Docs) in which the contribution facets are

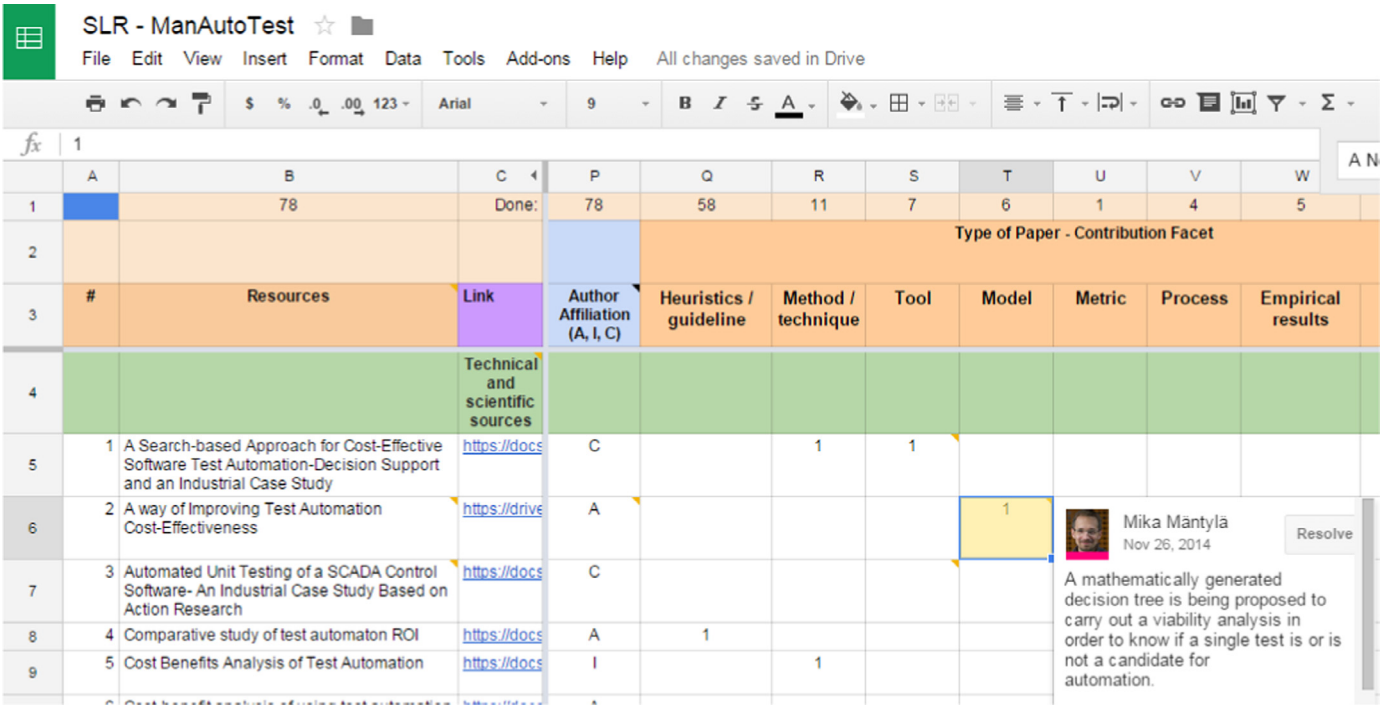


Fig. 8. A snapshot of the publicly-available spreadsheet hosted on Google Docs.

shown and a classification of 'Model' for the contribution facet of [Source 2] is shown, along with the verbatim copy/paste text from the source acting as the corresponding 'traceability' link.

During the analysis, each of the two researchers extracted and analyzed data from half of the sources (assigned to him), then peer reviewed the results of each other's analyses, and also did independent voting on either to include or exclude each source. In the case of disagreements, discussions were conducted. This was conducted to ensure quality and validity of our results.

To choose our method of synthesis for RQ 3 (factors considered in the when/what questions), we carefully reviewed the research synthesis guidelines in SE, e.g., [77–79], and also other SLRs which had conducted synthesis of results, e.g., [80,81]. According to [77], the key objective of research synthesis is to evaluate the included studies for heterogeneity and select appropriate methods for integrating or providing interpretive explanations about them [82]. If the primary studies are similar enough with respect to interventions and quantitative outcome variables, it may be possible to synthesize them by meta-analysis, which uses statistical methods to combine effect sizes. However, in SE in general and in our focused domain in particular, primary studies are often too heterogeneous to permit a statistical summary. Especially for qualitative and mixed methods studies, different methods of research synthesis, e.g., thematic analysis and narrative synthesis are required [77].

The factors and their categorization presented in the paper were the results of a formal and systematic synthesis done collaboratively between the two researchers following a systematic qualitative data analysis approach [83]. We had some pre-defined factors (based on our past knowledge of the area), namely "regression testing", "maturity of SUT" and "ROI". During the process, we found out that our pre-determined list of factors was greatly limiting, thus, the rest of the factors emerged from the sources, by conducting "open" and "axial coding" [83]. The creation of the new factors in the "coding" phase was an iterative and interactive process in which both researchers participated. Basically, we first collected all the factors affecting W&W questions from the sources. Then we aimed at finding factors that would accurately represent all the extracted items but at the same time not be too detailed so that it would still provide a useful overview, i.e., we chose the most suitable level of "abstraction" as recommended by qualitative data analysis guidelines [83].

Fig. 9 shows the phases of qualitative data extraction for the factors, in which the process started from a list of pre-defined categories: maturity of SUT, stability of test cases, 'cost, benefit, ROI, and need for regression testing) and a large number of 'raw' factors phrased under the 'other' category. By an iterative process, those phrases were qualitatively coded (by applying 'axial' and 'open' coding approaches [83]) to yield the final result, i.e., a set of cohesive well-grouped set of factors (to be discussed in Section 6.3).

6. Results

Results of our study are presented from Section 6.1 to 6.4.

6.1. RQ 1-mapping by contribution and research facets

6.1.1. RQ 1.1-mapping of studies by contribution facet

Fig. 10 shows the annual cumulative breakdown and the total number of primary studies by contribution facet types. Exact study references have also been provided under the figure. The top three contribution facets are: heuristics and guidelines, methods (also referred to as technique or approach), and metrics, which have appeared in 41 studies (68%), 39 studies (65%), and 12 studies (20%), respectively. Note that most of the time-trend charts in this study such as the one shown in Fig. 10 are cumulative stack charts, showing the cumulative number of studies in each year, as accumulated from previous years.

Further note that since many studies were classified under more than one contribution facet, the stack chart of Fig. 10 cannot be used as the annual trend of number of studies (that trend was presented in Section 4.3.1). We can see that across different years, different contribution facets have been studied, and no clear trend change is observable.

58 sources proposed heuristics and guidelines. As discussed in Section 5.2, heuristics and guidelines in this context are

AH	AI	AJ	AK	AL
39	38	7	44	57
FACTORS considered (in when/what to automate)				
Stability of SUT	ROI	Test stability	Regression testing	Other
	1			
			1	Test reuse, revelevance, automation effort, resource (money, hw), manual complexity, automation tool quality, test portability, manual exec effort
1		stable enviroment and data	1	design the test case first, test engineer's skill, first manual then automatte, SUT known, risk and complexity low
	1			reduce costs and improve quality because of more testing in less time, but it causes new costs in, for example, implementation Generic and independent products facilitate and customized and complex products hinder testing automation High reusability facilitates and low reusability hinders testing automation
1			1	Easiness to automate, boringness of the test, manual test must exits first, Automate non-time dependent tests, Automate tests that have been written:
	1			

(a): initial phase of qualitative coding

SLR - ManAutoTest

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

fx High reusability facilitates and low reusability hinders testing automation

	A	B	C	D	E	F	G
1	78	Done:	78	32	17	14	18
2	Paper Title	Link	Who	Other			
3	Technical and scientific sources	.			Test reuse/ repeatability	Manual test effort	automatability
13	Pragmatic approach to software test automation	0B6dKdxaNj	v	design the test case first, first manual then automatte, SUT known, low test risk and complexity			
14	Software Test Automation in Practice: Empirical Observations	0B6dKdxaNj	v		High reusability facilitates and low reusability hinders testing automation		consider automation costs
15	Software test automation–Developing an infrastructure designed for success	ilt/files/article	m			boringness of the test	automatability
16	Surviving the Top 10 Challenges of Software Test Automation	0B6dKdxaNj	v		Test repeatability		automatability

(b): intermediate phase of qualitative coding

SLR - ManAutoTest

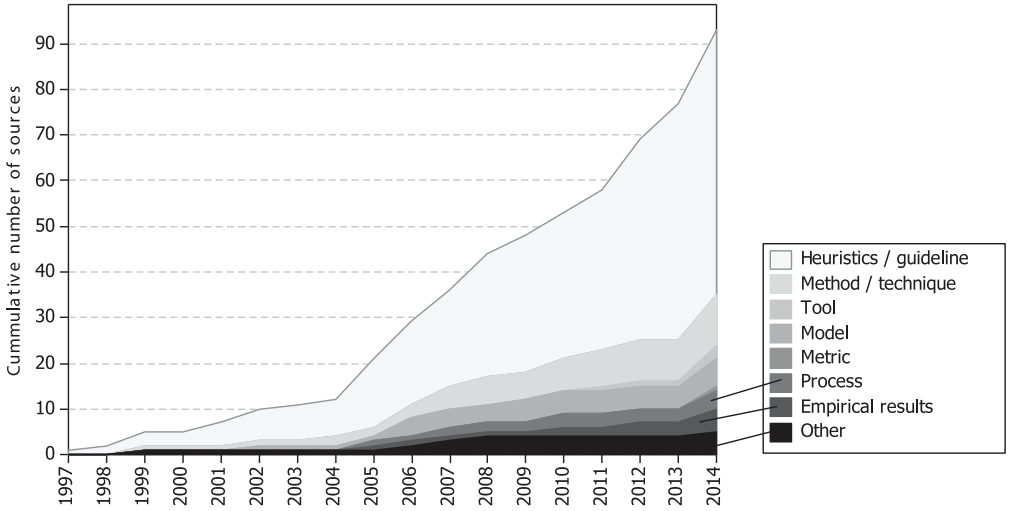
File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

fx Decision support

	A	B	C	AJ	AK	AL	AM	AN
1		78	Done:	39	6	44	30	17
2				Factors considered (in when/what to automate)				
3	#	Resources	Link	Stability of SUT	Other SUT aspects	Need for regression testing	Test Type	Test reuse/ repeatability
14	10	Pragmatic approach to software test automation	https://docs	1		1	low test risk and complexity	
15	11	Software Test Automation in Practice: Empirical Observations	https://docs		SUT features (Generic and independent products facilitate and customized and complex products hinder testing automation)			High reusability facilitates and low reusability hinders testing automation
16	12	Software test automation–Developing an infrastructure designed for success	https://drive	1		1	Automate tests that have been written (manual test must exits first), Automate tests with no timing issues	
17	13	Surviving the Top 10 Challenges of Software Test Automation	https://docs		SUT criticality/risk			Test repeatability
18	14	The Return on Investment (ROI) of Test Automation	https://docs					
	15	The When & How of Test Automation	https://docs	1				

(c): the final result after qualitative coding

Fig. 9. Phases of qualitative data extraction for factors considered for deciding when/what to automate.



Contribution facet types	Number of sources	References
Heuristics / guideline	58	Too many to be listed. Refer to the spreadsheet (http://goo.gl/zwY1sj)
Method / technique	11	[Sources 1, 5, 7, 8, 17, 43, 51, 56, 57, 61, 62]
Tool	7	[Sources 1, 17, 61, 75 -78]
Model	6	[Sources 2, 7, 17, 21, 22, 38]
Metric	1	[Sources 6]
Process	4	[Sources 15, 17, 47, 51]
Empirical results	5	[Sources 3, 6, 9, 11, 16]
Other	5	[Sources 5, 7, 31, 48, 62]

Fig. 10. Mapping of studies by contribution facet.

informal recommendations on W&W to automate and are usually less mature than methods and techniques which have been categorized differently. 11 source proposed methods /techniques addressing the W&W questions. For example, [Source 1] proposed a search-based approach (using genetic algorithms) for cost-effective software test automation-decision support and a supporting industrial case study. We found also four online tools for supporting the when and what questions in software test automation [Sources 75, 76, 78, 77]. Those tools will be briefly reviewed when answering RQ 4 in Section 6.4. Six sources proposed models in support of the W&W questions. For example, [Source 2] proposed a mathematically-generated decision tree to carry out a viability analysis in order to know if a test case is or is not a candidate for automation. [Source 7] proposed an opportunity-cost model. In terms of metrics used to answer the W&W questions, many studies used the conventional ROI metric. Four sources [Source 15, 17, 47, 51] proposed processes addressing the W&W issues. Fig. 11 shows the reference process adopted from [Source 17]. The contributions of five sources [Source 3, 6, 9, 11, 16] related to our context were only empirical results and could not be categorized under the other contributions category types. [Source 6] was a master thesis which presented a case study on the profitability of test automation in the development of embedded software in an industrial setting. Five sources [Sources 5, 7, 31, 48, 62] contributed 'other' types of contributions. [Source 5] presented a set of falsely expected benefits (myths) in test automation such as: *"All tests will be automated: This isn't practical or desirable.*

Next, we wanted to compare the ratio of different contribution types in this area with one other representative sub-area in testing. Fig. 12 shows the mapping of studies by contribution facet in the area of software test-code engineering (STCE) and in this paper. The STCE data is taken from another SM study [70]. Except for the case of large number of heuristics presented by sources in the current MLR which is due to having large number of sources form the grey literature, the other ratios are quite comparable.

6.1.2. RQ 1.2-mapping of studies by research facet

Based on the scheme described in Section 5, we classified the studies into six research-facet categories. Fig. 13 shows the classification of the all sources according to the type of research method they have followed and reported. Recall that, for the research facet type, each study could be categorized under one or more categories.

A large ratio of sources fall under experience- and opinion-based categories, which is again due to having large number of sources form the grey literature in our pool. There are only one rigorous empirical study (evaluation research) [Source 16] and six weak empirical studies (validation research) [Sources 1–3, 6, 17, 61]. In [Source 11] (categorized under 'Other' in Fig. 13), a qualitative survey-based study on 55 test specialists in 12 selected software development organizations was conducted. We interpret the low number of sources using Evaluation Research (strong empirical study) due to two possibilities: (1) low since mainly researcher conduct strong empirical studies, there has been low attention of the research community in this domain; (2) complexity of conducting strong empirical studies in this area.

We also thought it would be useful to compare the research facet breakdown of this SM to four other SM studies in SE that have utilized the same classification [60–62,70]: web application testing [60], GUI testing [61], software test-code engineering (STCE) [70], and Dynamic Software Product Lines (DSPL) [62]. The comparison is visualized in Fig. 14, and it shows that the shares of research facets in the other four SM studies are quite similar to each other with solution proposal, validation research and

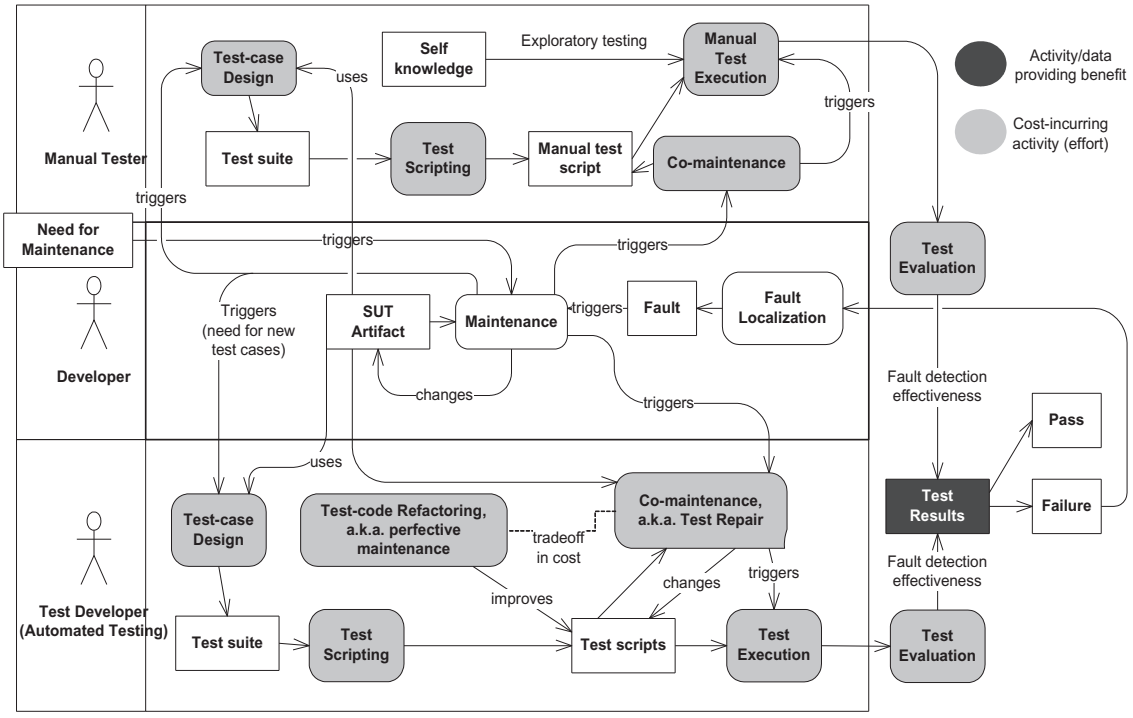


Fig. 11. Software testing reference process (adopted from [Source 17]).

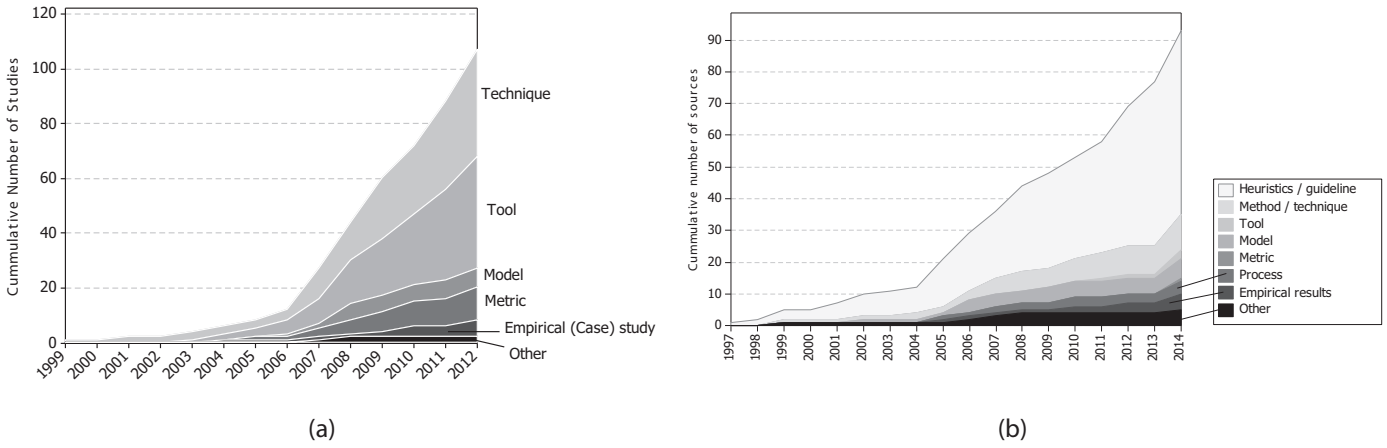


Fig. 12. Mapping of studies by contribution facet in the area of software test-code engineering (a) (data is from another SM study [70]), and in this study (b).

evaluation research having the largest shares. However, in this paper, the ratios of experience-based and opinion sources are high since a large number of gray literatures (e.g., blog posts) have been included.

6.2. RQ 2-factors considered for deciding when/what to automate

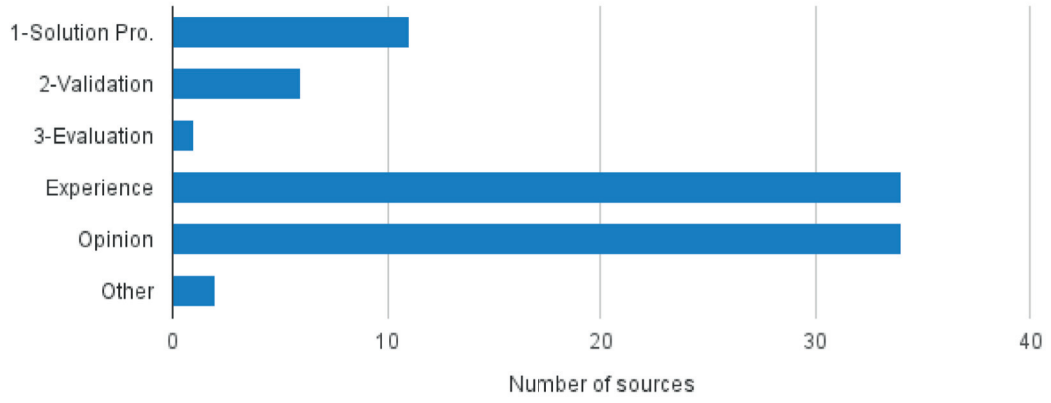
In this Section, we present the qualitative classification of the factors to consider when deciding on when and what to automate. We think this Section has the highest value for practitioners whereas other sections are meant more towards academic audience.

6.2.1. A simplified view of software test automation

In test automation, there are four basic components (see Fig. 15): testers (test engineers), (test automation) tool, test cases and test suites, and system under test (SUT). Test engineers interact with the test automation tools and develop the test cases. The test cases are then executed using the chosen test automation tool. The tests exercise the SUT and the tool provides the test reports for humans to interpret. According to the pool of sources reviewed in our study, success or failures of test automation depend on all of these factors and on their interactions, as we discuss in the next sections. Additionally, we found a number of factors and have placed them under a category called “cross-cutting factors” (labeled as #5 in Fig. 15) which spans across the other four categories. Examples of such cross-cutting factors are economic factors, automatability of testing, manual test effort, and development process, which will be discussed later in the paper.

6.2.2. Categorization of factors impacting when and what to automate

During our review, we extracted the factors mentioned in each source and classified them into 15 types of factors under the five categories as shown in Fig. 4. The frequency of different factor types as discussed in the sources is shown in Fig. 2 to illustrate the level of attention on each factor. Note that, these numbers are not meant to be indicative of importance as what is important varies



Research facet types	Number of sources	References
Solution proposal	11	Refer to the spreadsheet (http://goo.gl/zwY1sj)
Validation research	6	[Sources 1 -3, 6, 17, 61]
Evaluation research	1	[Source 16]
Experience studies	34	Refer to the spreadsheet
Opinion studies	34	Refer to the spreadsheet
Other	1	[Source 11]

Fig. 13. Mapping of studies by research facet.

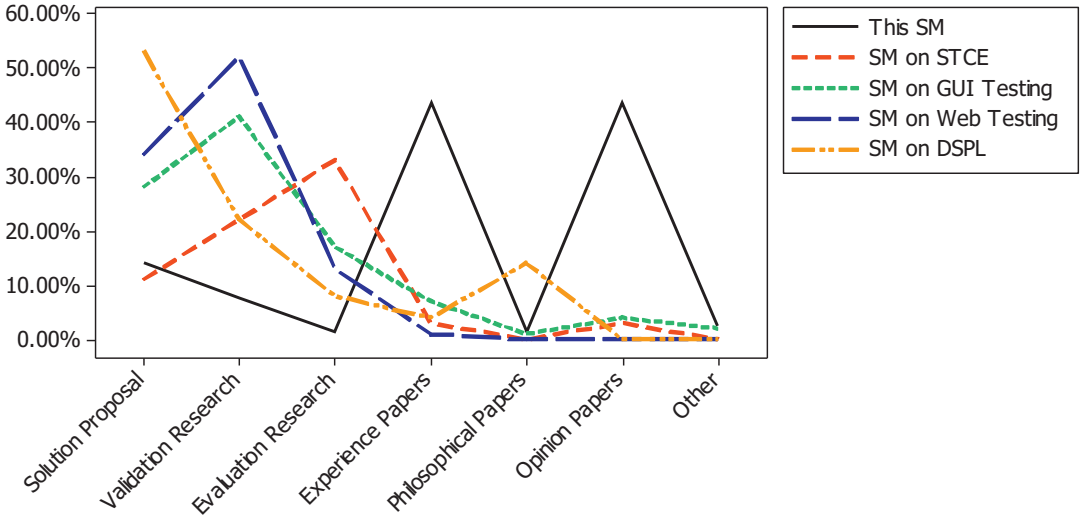


Fig. 14. Comparing the research facet breakdown of this SM to other representative SM studies [60–62,70].

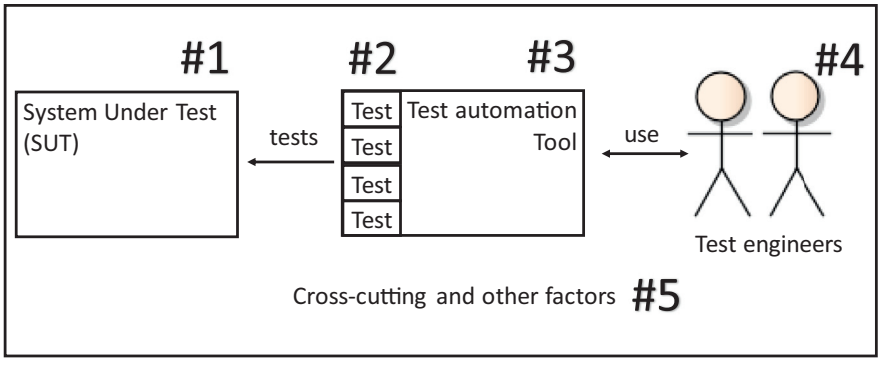


Fig. 15. A simplified view of software test automation.

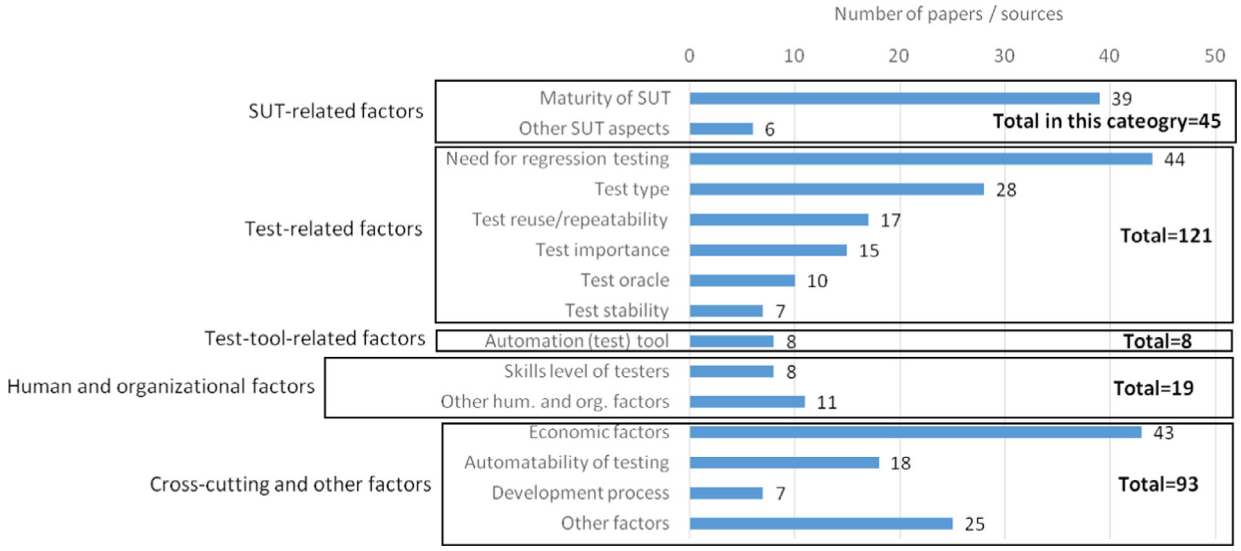


Fig. 16. Factors considered when deciding when to automate testing and what part(s) of the SUT should be automated.

from case to case. For example, eight sources that we had in the pool mentioned that skills level of testers shall be carefully considered when deciding when and what to automate. Note that a single source often mentioned several factors, and thus some of the totals in the figure, e.g., 121 for the test-related factors, are more than 78 (number of the sources in the pool).

As our review revealed, the two questions "what to automate" and "when to automate" are closely related. The "what to automate?" question was interpreted in our sources as what test cases to automate, e.g. what test types, what test cases, and what features. The "when to automate?" question was interpreted in our sources in two ways: (1) when to start automation during a software project life-cycle, and (2) under what conditions, it is preferable to implement a test case in automated fashion rather than manual, e.g. when one has technically skilled individuals, when automation feasibility is high, when suitable tests are found. The second interpretation of "when" has very high overlap with the "what to automate?" question. Thus, our decision to analyze them globally was based on our source articles that we reviewed in this investigation, i.e., we wanted to be as compatible with them as much as possible. Next we present our findings with respect to each of the five categories shown in Fig. 2.

6.2.3. SUT-related factors

Properties of the system under test (SUT) have major impacts on the automation decisions (Fig. 16). 45 sources discussed factors in this category. If the SUT is not mature enough (discussed in 39 sources), e.g., due to new features being re-implemented or major ongoing changes, there will be a major negative impact on automated tests (also called "broken tests" [84]) as the number of false-positive defect reports from automated tests would increase and the effort needed on fixing (repairing) automated tests and analyzing false-positive defects would reduce the benefits of automated testing. In general, the effort to keep tests up-to-date with latest changes in the SUT, often referred as test repair [84] or test co-maintenance, is a major concern in the area of automated testing. Some direct quotes from the sources in this category of factors are:

- "*Automation fails when the current application has unstable design*": in a paper written by test engineers at an India-based firm named United Health Group [Source 10]
- "*Automate tests for stable applications*": in a paper on developing an infrastructure designed for test automation success [Source 12]
- "*Selection of (the right) use cases before starting to automate them can prevent large amounts of rework in terms of test maintenance activities in the project.*": in an industrial case study [85] in which the first author was involved [Source 1]
- "*The SUT must have reached a certain level of maturity for VGT (visual GUI testing) to be applicable*": in a paper on practical challenges of visual GUI testing [Source 16]

Other SUT factors such as length of lifetime, high customizability, complexity, and dependence on 3rd party applications are factors which would impact the decision to automate testing.

- "*Test automation is an effective solution when the life of the application released is long*": in a white paper by Infosys [Source 48]
- "*Main Application has lot of interdependency with other Applications which in turn cannot be automated.*": in a presentation by IBM engineers [Source 50]

Summary: To make the proper decision w.r.t. the SUT-related factors, one need to take into account the SUT's maturity and stability and to predict whether there would be many future changes forthcoming that might require major test co-maintenance effort.

6.2.4. Test-related factors

The characteristics of test cases and test suites (Fig. 16) also impact the automation decisions, i.e., what tests (not) to automate? We found that a need for regression testing was the most frequently mentioned factor of test automation decisions (mentioned in 44 sources, more than half of the entire pool). It seems that the importance of this factor in making the right decision (when and what) will continue to increase as the continuous and rapid releases of software become more frequent.

- "*Once automated, regression tests can be efficient, and effective. Accordingly, ABB decided to focus its attempt to establish automated testing in build regression tests, which are most suited for automation and where the benefits could be attained*": in an industrial experience report by two engineers of the ABB Corporation [Source 8]

- In a blog post entitled "*For those of you dreaming the 100% automation dream...please wake up!*" [Source 37], a Microsoft engineer quotes from another test practitioner named James Hancock as follows: "*James Hancock has estimated that an automated test must run 15-17 times to break even on the cost of developing that test*".
- There are also extreme suggestions such as: "if you are going to run a test more than once, it should be automated" [Source 39].

28 sources considered test types as a factor for decision making. Certain types of tests are good candidates for test automation while others are not. For example, both researchers and practitioners reported that performance and load testing are often very hard to be performed manually and they should normally be automated.

- *"When to automate testing? … Load and performance testing: simulate hundreds or even thousands of virtual users across multiple devices"*: in a blog post by Borland Corporation [Source 41]

Similarly, tests that human dislike to perform were suggested for automation. In the other hand, candidates for manual testing are related to user experience (UX) and usability testing, but also tests that are not stable due to timing issues for example (e.g., in real-time systems). Such tests offer little payback for automation.

Test reuse and repeatability (mentioned in 17 sources) are other factors which are closely-related to regression testing. They can refer to cases where the same test can be reused as a part of another test, e.g., a login test must pass in a web-based application before other tests can be executed. The number of environments to test a SUT in usually increases test repeatability, e.g., when testing Android applications, one needs to repeat the same test in different Android phone models.

- "*When to automate testing? Testing across multiple OS platforms and multi lingual sites*": in a blog post by Borland Corporation [Source 41].

As another factor, test importance was mentioned in 15 sources and stems from two factors. Firstly, if a given test targets the important functionalities of a SUT that is highly critical for user satisfaction, then this makes the test more important and it is thus encouraged to be automated. The other source for test importance is the test's likelihood of revealing defects.

- "*The number of bugs the test case is supposed to (or can) find is another point to be considered [in deciding to automate]*", mentioned in a technical paper by the Brazilian Center of Advanced Studies and Systems [Source 2]

The existence and stability of test oracles (discussed in 10 sources), i.e., the mechanism of how defects are detected, are also important. Unpredictable outcomes that require human judgment as oracles may not be worth automating. On the other hand, if the test oracle is stable and predictable, then test automation is recommended as humans do not excel in vigilance.

- *"Non-deterministic results may make automatic testing difficult regardless of API or UI"*: in a technical paper by a test architect at Microsoft [Source 64]
- "*Automation is all about predictability. If you cannot express the precise inputs and expected outputs, you cannot automate a test*" [Source 38]

Test stability is another factor in this category since if (the logic of) a test is not stable; automating it is not a good idea [Source 60]. Last but not the least, other candidates for automation is tests whose inputs are predictable [Source 38].

Summary: Introduction of test automation often increases cost for creating tests, however the cost of re-running a test decreases. Thus, understanding the number of needed test re-runs and the effort of test creation and maintenance are important. Additionally, some test types such as performance tests are better targets for test automation compared to others such as user experience tests. Finally, the sources of test oracles and their stability are also critical issues to consider.

6.2.5. Test tool-related factors

Quality and adaption of proper and suitable test automation tools are also important factors for the when and what to automate questions, which were highlighted by 8 sources (Fig. 16).

- *"Is the test tool being utilized for automated testing capable of interacting with all of the necessary attributes of the feature for test purposes? (For example, can it interact with it as well as the users will be able to? Can we capture all of the necessary data from the GUI and child objects?)":* as mentioned in a book by two industry consultants [Source 22].

In general, it appears that the lower-level testing tools and framework (e.g., in unit level) are more of the same type and mechanism, since almost the entire test tools are based on the family of the xUnit framework. However, at the system testing level, the tools are very diverse and highly dependent on the application domain, for example, for automated testing of web-based applications, the Selenium tool is quite popular and while for testing telecommunications systems, tools based on the TTCN (Testing and Test Control Notation) are widespread. If a proper test automation tool for a particular situation or need is lacking and it is not possible to develop such a tool in-house, then one is advised not to progress with test automation. Several papers and sources acknowledged the need to do proper tool selection and some sources present a process how to do it.

- *"There are hundreds of automation tools available in the market. A careful effort has to go into deciding which tool would be most suitable for automating the testing of your product/application."*: in a technical paper by Infosys [Source 15]
- *"The automation tool must be very carefully chosen before the test automation process begins. ... Complex functions that might not have the necessary trustworthiness because of an automation tool's dependencies must be well considered before being created. This seems like simple advice, but it's a very difficult issue to solve: wrong results reported by the tool."*: in a technical paper [Source 2]

Summary: One needs to understand the test tool and its compatibly with the SUT, and the business model of the tool vendor. The future direction of the tool is also important as changing the testing tool to another in the midst of a project is usually non-trivial. Popular open-source tools are often good options as they have a low cost (e.g., only training, etc.), do not have the risk of a single tool vendor (such as increasing prices, sudden stopping of tool development, etc.), and large user bases which can be seen as the insurance of the tools' future sustainability and success.

6.2.6. Human and organizational factors

Human and organizational factors also affect test automation decision (Fig. 16). Test automation requires different (and often additional) skills than manual testing. If the testers' team lacks programming skills, introducing test automation to such staff requires training or runs a high risk of failure (discussed in 8 sources). Thus, when competencies are lacking and there are no resources available for the training that is needed to acquire the skills, then it might be better not to automate. Software testers may also view

test automation as a threat (to their job security) and resist it [Source 68].

- "*…different skills are required to implement an effective automated testing program from those required for manual testing*": in a book by three test engineers working in a test automation firm [Source 21]
- *"Testers/QA organizations may be comfortable and experienced with manual testing and feel threatened by automation"*: [Source 68]

In addition to testers' skills, a number of other human and organizational factors were also discussed in 11 sources, e.g., organizational maturity [Source 8], time and resource constraints [Source 50] and need for proper change management [Source 73]. A quote is as follows:

- "*An organization that is mature enough to handle process improvements in a structured manner is a prerequisite to establishment of automated testing. If not mature enough, it is inevitable that the organization will yield even more chaos when performing the introduction*": an industrial experience report by two engineers of the ABB Corporation [Source 8]

Test automation typically requires a high initial investment before the benefits start appearing [86]. Thus, tight schedule might prevent the introduction of test automation.

- *"If you are on a fast-track project where the project management has a very tight delivery schedule, you can pretty much forget about automation unless adequate time has been allocated specifically for it"*: in a book by two industry consultants [Source 20]

Summary: Introducing test automation is also an organizational change, a management issue which is prone to failures and has a large body of consulting literature giving advice on it. To successfully consider this factor in decision making, one needs to understand organizational politics, current company competences, how to organize training, and perhaps, most importantly to have soft skills to get people to accept and even to get excited about test automation.

6.2.7. Cross-cutting and other factors

We also identified several factors that were cross-cutting, i.e., they are applicable in the context of more than one group. Those factors were: (1) economic factors, (2) automatability of testing, and (3) development process. Also, a group of "other" factors which could not be placed under any of the above categories was identified, which are discussed next.

Factors under the economic factors category relate mainly to the cost and effort tradeoffs between the manual and automated testing. In total, as shown in Fig. 2, 43 sources mentioned factors that fall under this category. Here, the emphasis is on the effort spent and benefits gained through automated or manual testing, e.g., Return-Of-Investment (ROI) calculations. A simple example of the test automation ROI chart is depicted in Fig. 17. Although a ROI perspective as shown in this figure may look too simplistic for some readers, many of the existing decision-support tools in this area (e.g., the one by the IBM [Source 77]) are based on this simple ROI model (see the online spreadsheet http://goo.gl/zwY1sj for details).

Common wisdom suggests that test automation is more economical compared to manual testing if one has to perform several iterations of testing [86]. Usually, the upfront cost of automation is more than manual testing but costs of automated test execution are less than manual testing, especially if testing will be repeated many times. Interestingly, some practitioners presented counter arguments to ROI calculations as well.

- "*if you have a hard time convincing someone that it needs to be automated and if you feel you need an equation to see the long-term advantages of automation, don't bother automating it*" [86]: an online article by a software testing consultant [Source 52]
- "*Automate all things that offer immediate returns*": a white-paper by a software testing consultant [Source 57]

Automatability of testing, i.e., how easy it is to test the SUT in an automated manner, is another factor under this group (mentioned in 18 sources). Automatability is not only a property of SUT but it is also affected by the test automation tool(s) and human skill level for example. That is why we have considered it a "cross-cutting" factor.

- *"For example, getting the user interface design team to change editable fields into non-editable pull-down fields wherever possible—such as on date and time fields—can reduce the size of the potential user input validation test set dramatically and help automation efforts."*: in the guide to the International Software Testing Qualifications Board (ISTQB)'s Advanced Test Manager certification [Source 18]

The choice of development process as another factor to impact the decision was addressed in seven sources. Only two papers addressed the Agile vs. Waterfall process in particular:

- "*Should you automate every (Agile) project? … I don't think it makes sense for a project that's only 2-3 sprints long*": an online article by Director of Quality Assurance at eSecuritel [Source 28]
- *"XP testers have to drive in the fast lane … you need a lightweight automated test design and lightweight test tools"*: in a paper by two test engineers [Source 55]

The number of releases, an important aspect of software development process, was an input considered in three of the four decision-support tools offering ROI computations which were in our pool. Higher release frequency increased the benefits of test automation. More general guidance with respect to development process was also given:

- *"From a process perspective, [we] need to determine how test automation will fit into the system development process"*: in an online article by a senior test architect [Source 54]

We realize that some of the factors discussed in this article have implicit relationship to the question whether development process affects software automation decisions, e.g., test reuse and repeatability is usually applicable in Agile development processes in which frequent iterations of testing is conducted. On the other hand, in the Waterfall model, testing is a single phase towards the end of the project, the need for test automation is generally less, because in the beginning there is no code for which the test automation could be used to verify, the testing phase is often under time pressure, and there is less occurrence of continuous integration and delivery and thus less justification/need for test automation. Yet, such generalizations can be dangerous as the decision in this context are very "context-driven" and reminds us of the famous quote in software engineering: "*It depends!*".

25 of the total of 78 sources discussed other factors that could not be placed under any of the above categories and were quite sparse to deserve categories (clusters) of their own. We thus grouped them under the "Other" category in Fig. 16. A few of those factors (recommendations) that should be considered are:

- "*[Automate when] Scope of automation has been defined*": [Source 15]
- "*Complexity of the test environment*": [Source 27]

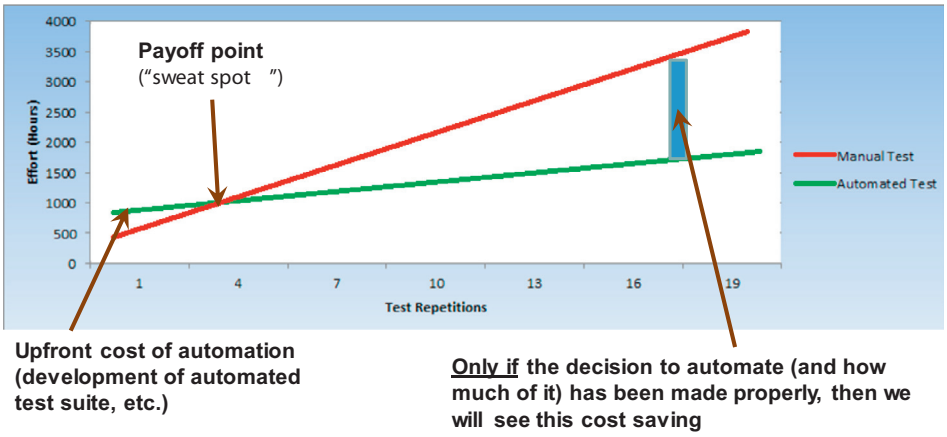


Fig. 17. An example chart of the test automation ROI supported by many industrial tools.

- "*The most common key program factors are: ... Quality objectives (defect escape velocity)*" [Source 54]
- "*[Need for more] coverage, productivity, accuracy*": [Source 68]

Summary: To successfully use cross-cutting factors in decision making, one needs to track testing with respect to cost (effort) and benefits which would also enable ROI calculation of automation. If the SUT or test cases have low automatability (how easy it is to test in an automated manner), automation success will be in jeopardy. The software development process and the release model chosen in the context of a project are also important factors to consider when evaluating the use of test automation.

6.3. RQ 3-tools proposed to support the when/what questions

We found also four online tools for supporting the when and what questions in software test automation. All the tools where in practice ROI calculators taking in to account various factors affecting software test automation costs and benefits similar to the factors presented in previous section. The tools were offered by companies focusing on general IT, software and consulting (IBM) [Source 75], software test automation solutions (Elbrus Ltd and Automated Testing Institute) [Source 76, 78], and software development and testing (GlobalNow IT Services) [Source 77].

In generally, the tools offered a limited view of factors affecting test automation (Refer to the spreadsheet http://goo.gl/zwY1sj for details). For example different tests were not considered, but rather all tests were seen as equally valuable and equally automatable. Also only one tool [source 75] considered training of testers. Three tools did not consider as a factor the amount of regression testing that was the most popular factor as discussed in the previous section, see Fig. 16. However, they provided a factor number of releases, which is one of the factors explaining need for regression testing. Surprisingly, we found that no tool listed the factor Maturity of SUT that the 3rd frequent factor in previous section. Thus, we think that classification of the previous section and the details of our spread sheet can be used build a much more accurately decision support and we currently in the process of doing so.

6.4. RQ 4-software systems under test or projects under study

6.4.1. RQ 4.1-Number of software systems or projects under analysis in each source

For each source, we studied, how many software systems or projects under analysis had been used in each source? We expected that a given paper or article would apply the proposed idea to at least one system to show its effectiveness. We found that only 28 out of 78 (36%) sources had at least one system which they had investigated. Again, we suspected that the number is caused by a high number of grey literatures, see Fig. 6. We found that in grey literature only 13 sources out of 52 (25%) while in formally published works, books and scientific articles, 15 out of 26 sources (62%) had studied at least one system. Furthermore, we found that, in the majority of the papers, the number of systems or projects studied was one, see Fig. 18 (17/27) and in total there were only four papers where the number of systems studied were four or higher.

6.4.2. RQ 4.2-domains and types of the software systems or projects under analysis

The domains of the software systems and projects under analysis varied across the sources in the pool, and we could not find a particular domain that would dominate in our area. The wide range of domains found in the sources were for example web applications, mobile applications, office software, finance, control systems (e.g.,. elevators), and pharmaceutical.

With respect system types, i.e. real open-source, real commercial, or experimental toy systems, we found that real commercial systems were studied in 17 cases whereas experimental ('toy') systems were utilized in 11 sources. No sources utilized real open-source systems such as Linux kernel and Mozilla Firefox. Surprisingly, six of the sources using toy-based examples came from grey literature whereas five came from the formally-published literature. This means that the formal literature contributed 12 real-world sources while the grey literature contributed only five. Despite the large numbers of grey literature, they provided surprisingly little real-world evidence, and even that evidence was often reported in a less-rigorous manner.

6.4.3. RQ 4.3-measurements to support the when/what questions

Finally, we were interested in identifying the paper that had actually collected empirical evidence in the form of quantitative measurements with respect to questions on when and what to automate. Obviously this question is only relevant with the 17 sources that had analyzed real world systems. We found that there were 11 cases reporting quantitative benefits. The ROI was reported in four sources and it varied between 40% and 3200% and the number of test executions or iteration required was reported in two sources that reported 13 [Source 16] and 22 [Source 46] iterations of testing. However, such numbers cannot be used for any type of generalizations as the factors affecting the when and what to automate and the economics of it vary largely between cases as illustrated by our factor taxonomy in Fig. 6.

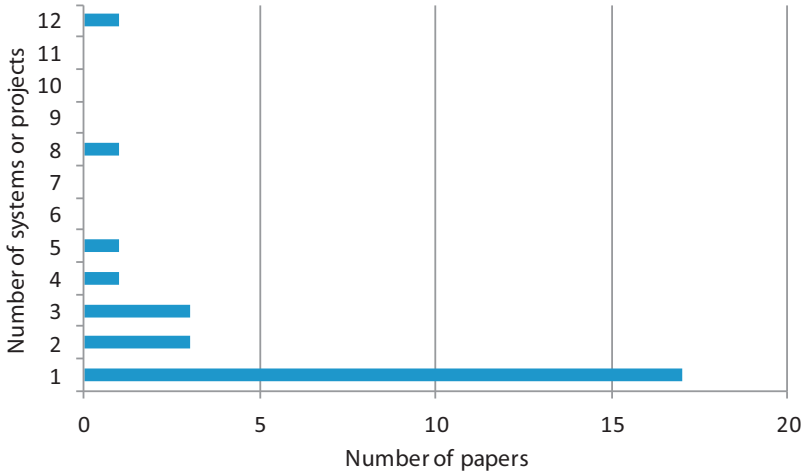


Fig. 18. Number of software systems or projects under analysis in each source.

7. Discussions

Section 7.1 discusses the summary of findings, and implications of our MLR. Section 7.2 discusses potential threats to the validity of our study and steps we have taken to minimize or mitigate them.

7.1. Summary of findings

Software testing practitioners have a need for test-automation decision-support or criteria. We think this is visible from the number of Google hits produced by our search strings, see Section 4.1, and from the fact that two thirds of the literature of this review paper are grey literature published by practitioners, see Fig. 6. Additionally, according to the authors best knowledge no systematic literature reviews have been performed on the questions when to and what to automate in software testing. These drivers motivated us to perform a MLR study in this area. Our MLR study, answered four review questions (RQ1 to RQ 4). Below we summarize the results of the RQs and discuss the findings for the research community and practitioners.

7.1.1. RQ 1-mapping by contribution and research facets

RQ1 mapped sources by contribution and research facets. We found that the majority of the papers (58) of the weakest decision support as they only provide heuristics or guidelines. A more rigorous decision support is offered by 11 papers whose contribution we classified as a method or a technique. Seven papers propose a tool for deciding when or what to automate. Note that some papers that offered techniques also proposed a tool support their techniques. Four papers present a process on how to decide about test automation and six papers provided models on the topic. As a summary, we can state that there seems to be more room methods, tools, processes and models in this area. There already are plenty of heuristics which represent the weakest form of decision support.

We identified a similar trend when mapping sourcing with respect to research contribution. A majority of the papers were based on opinion (34), which is the weakest form of evidence, experience (34), which is the second weakest. We found only six studied validation and one study of evaluation research that present more rigorous empirical evidence of the decision support. Comparison to other SLRs showed that this is not typical, see Fig. 13. We think also this implies the practical importance of the topic but the general lack of academic interest in presenting systematic decision-making support for software test automation.

7.1.2. RQ 2-factors considered for deciding when/what to automate

We synthesized all the factors affecting test automation decision by using qualitative data analysis guidelines [83], as described in Section 5.2. We identified 15 factors that we furthermore classified to five groups, namely SUT-related factors, Test-related factors, Test-tool-related factors, Human and Organizational factors, and Cross-cutting and other factors. We found that the Need for regression testing was the most frequently mentioned factor with 44 sources followed closely by Economic factors (43) and Maturity of SUT (39).

The frequencies listed are, however, only illustrative and should not be used as of indication of the importance of each factor. We think that each particular case where test automation takes place should consider and rank the factors based on importance in that particular context. For example, Skill level of testers was mentioned in eight sources only. Still, for some context skill level of testers might the number one hurdle while in some others, e.g., ones that have already done plenty of test automation, would need to pay only little attention to that factor. Thus, the classification should serve as a checklist for practitioners making test automation decisions rather than a normative list giving absolute priorities. For academics, it can offer ideas for future studies.

7.1.3. RQ 3-tools proposed to support the when/what questions

We find four different online tools that were available openly or by completing free registration. These provided by companies offering test automation and other types IT services such as IBM. We found that the tools considered an inadequate list of topics for test automation decision support when compared with our list of factors. This highlights the need to build better decision support that is openly available to the practitioners.

7.1.4. RQ 4-software systems under test or projects under study

We found that only 64% of the papers had not studied any software systems or projects to justify their advice on when to automate and what to automate in software testing. It was even more surprising that the complete lack of systems analyzed was more frequent (75%) in the gray literature stemming mainly from industrial sources in comparison to formally published literature (38%) published by both academics and industry practitioners. This suggests that requirements placed on formal publishing actually increase the amount of empirical evidence in software engineering. On the other hand, many gray literature sources were opinion or experience based, but failed to provide examples. Thus, the sources of knowledge, i.e. the epistemological source, were not formally identified. For example, when an author was stating “I think X”

the author could have had experience or even data from several test automation projects or the X could be based on what he had just heard recently from a colleague as a rough opinion.

Further analyses showed that the software system or project under study came from various different domains, e.g. web, mobile, embedded. With respect to types of system we found that 17 papers had analyzed commercial closed source systems while 11 papers had analyzed small toy or experimental systems. We found that no studies had analyzed real world open source systems. This was rather surprising given the popularity of analyzing the open source systems in research communities focusing for example on Mining Software Repositories (MSR) or Free and Open Source (FLOSS).

Finally, we studied the amount of sourcing giving quantitative measurement evidence on the benefits and drawbacks of software test automation of the real commercial system, thus, we excluded the toy systems for this part. We found that only 11 sources (14%) provided quantitative evidence of the topic. Thus, a lack of high quality real world studies of this topic exists.

7.2. Implications to practice and a checklist to support decision-making

In summary, our study aimed at benefitting practitioners by presenting a single source which synthesizes and summarizes all the approaches, guidelines and experience-based opinions w.r.t. the two W's of automated software testing (what and when), a need that many practitioners have personally expressed to us in our industry-academia interactions, e.g., [13–16]. Our study is also a contribution to the research community by capturing the state-of-the-art and –practice in test-automation decision-support and also by pointing out research gaps and needs in this active area. To make our results more usable for practitioners, we have developed and present in Table 3 a draft checklist that practitioners can use when assessing what and when to automate questions. The importance of each item in the checklist will vary case-by-case. Therefore, it is important that the users evaluate the importance of each claim in their own context. Often times, such an advice is given by consultants and they also appear in practitioners' text-books. We should note to the reader that the checklist represents our expert view based on the MLR results presented in this paper. It is not a scientifically-validated instrument as the usefulness of the checklist has not been empirically validated yet nor do we have baselines data to show industry averages. Both of these steps are good topics for future studies.

7.3. Identifying and addressing potential threats to validity

Based on guidelines for performing systematic literature review and mapping studies [63,64,66] and also based on our previous experience [6,43,70], we systematically identified and carefully addressed potential threats to validity of our study by taking steps to minimize or mitigate them. We discuss next the potential validity threats in the context of the four types of validity threats based on a standard checklist adopted from [87].

7.3.1. Internal validity

The systematic approach that has been utilized for source selection is described above. 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. In order to mitigate risk of finding all relevant studies, formal searching using defined keywords was conducted. Therefore, we believe that adequate and inclusive basis has been collected for this study and if there are missing sources, the rate will be negligible.

Applying inclusion/exclusion criteria can suffer from researchers' judgment and experience. Personal bias could be introduced during this process. As pointed out in Section 4.1 both researchers independently searched for the literature which already consisted of excluding papers that did pass the inclusion criteria. This could have potentially excluded some that should have been included. However, this problem is partially mitigated as both authors can be considered experts in software engineering research and software testing, e.g. both have permanent professor positions and both had completed their PhD's over 9 and 5 years, respectively, before conducting this study. Had the searching been done by fresh PhD students or even MSc students, then pair work would have been recommended in all phases to substitute the lack of subject matter expertise. For example, studies on pair programming [88] have shown that junior programmers receive the most benefit of pair work. We think similar logic works for research as well. Furthermore, to minimize this type of bias, joint voting was applied after the initial source inclusion and only sources passing the joint voting were selected for this study.

7.3.2. Construct validity

Construct validity is concerned with issues that to what extent the object of study truly represents theory behind the study [87]. Threats related to this type of validity in this study were suitability of RQs and categorization scheme used for the data extraction.

Review questions are designed to cover our goal. Questions are answered according to a categorization scheme. For designing a good categorization scheme, we have adapted standard classifications from [65] and also have finalized the used schema through several iterations.

A threat to construct validity comes from the lack of empirical evidence in the primary studies. The majority of the grey literature was opinion or experience based. Since the grey literature represents the voice of the practitioners, we could assume that they would be speaking from personal experience with real industrial systems, though those systems were not explicitly mentioned/discussed in the sources. This would make the empirical bases more solid. On the other hand, it may be that some practitioners could simply be repeating the ideas/opinions that they had heard from other practitioners. Thus, as the source of knowledge was not typically revealed in grey literature, we are faced with an epistemological problem. We do not know, how we know, what we know. This is a serious limitation in the primary studies but fixing it is obviously not possible.

However, the fact that we are relying mostly on opinions and evidence does not mean our result is incorrect. If the same individuals who provided their opinion or experience papers would be interviewed or surveyed with a questionnaire of the same topic, it would be highly likely that the result would be the same. The only thing that would be different would be that the research method that used to collect the voice of the practitioners would be more rigorous.

7.3.3. Conclusion validity

Conclusion validity of a review study is concerned with reaching appropriate conclusions through rigorous and repeatable treatment. In order to ensure reliability of our treatments, the entire pool of primary sources are analyzed and the data was reviewed, extracted and synthesized by the two authors.

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.

Table 3
A checklist to support decision-making on whether to automate software testing. The "+" sign means the given situation favors test automation while the "−" sign suggest not to automate testing. The area weight is the number of sources for each factor in our study.

Category	Area (weight, i.e., num. of sources)	Situation	+/−
SUT-related factors	Maturity of SUT (39)	SUT or the targeted components will experience major modifications in the future.	−
		The interface through which the tests are conducted is unlikely to change.	+
	Other SUT aspects (6)	SUT is an application with a long life cycle.	+
		SUT is a generic system, i.e. not tailor made or heavily customized system.	+
		SUT is tightly integrated into other products, i.e. not independent.	−
		SUT is complex.	−
		SUT is mission critical.	+
Test-related factors	Need for regression testing (44)	Frequent regression testing is beneficial or essential.	+
	Test type (28)	Tests are performance and load tests.	+
		Tests are smoke and build verification tests.	+
		Tests are Unit tests.	+
		There are large number of test that are similar to each other.	+
		Tests require large amounts of data.	+
		Humans are likely to make errors when performing and evaluating these tests, e.g. tests require vigilance in execution.	+
		Computers are likely to make errors when performing and evaluating these tests, e.g. test execution is not deterministic.	−
	Test reuse/repeatability (17)	Tests can be reused part of other tests.	+
		Tests needs to be run in several hardware and software environments and configurations.	+
		The lifetime of the tests is high.	+
		The number of builds is high.	+
	Test importance (15)	Tests are likely to reveal defects, i.e. high risk areas.	+
		Tests cover the most important features, i.e. high importance areas.	+
	Test oracle (10)	Test results are deterministic.	+
		Test results require human judgement.	−
		Automated comparison will be fragile leading to many false positives.	−
	Test stability (7)	Tests are instable, e.g., due to timing. We must perform the test repeatedly and if it passes above a threshold we consider that the test passes.	+
		Tests are instable, e.g., due to timing. The results cannot be trusted at all.	−
Test-tool-related factors	Automation (test) tool (8)	We have experimented with the test automation tool we plan to use and the results are positive.	+
		A suitable test tool is available that fits our purpose.	+
		We have decided on which tool to use.	+
		We can afford the costs of the tool.	+
Human and organizational factors	Skills level of testers (8)	Our test engineers have adequate skills for test automation.	+
		We can afford to train our test engineers for test automation.	+
		We have expertise in the test automation approach and tool we have chosen.	+
	Other hum. and org. factors (11)	We are currently under a tight schedule and or budget pressure.	−
		We have organizational and top management support for test automation.	+
		There is a large change resistance against software test automation.	−
		We have the ability to influence or control the changes to SUT.	+
Cross-cutting and other factors	Economic factors (43)	There are economic benefits of test automation.	+
	Automatability of testing (18)	Tests are easy and straight forward to automate.	+
		Test results are ease to analyze automatically.	+
		Test automation will require a lot of maintenance effort.	−
	Development process (7)	Our software development process requires test automation to function efficiently, for example agile methods.	+
		We make several releases of our products.	+

7.3.4. External validity

External validity is concerned with to what extent the results of our study can be generalized. As described above, defined search terms in the source selection approach resulted in having primary sources all written in English language; studies written in other languages were excluded. The issue lies in whether our selected works can represent all types of literature in the area of study (when and what to automate). For these issues, we argue that relevant literature we selected in our pool contained sufficient information to represent the knowledge reported by other researchers and professionals.

Chapter 8 of [87] describes external validity as ability to generalize results to industrial contexts. As it can be seen from the collected data through study, in addition to academic studies, good proportion of industrial and collaborative works exists in our sources. This means that our inclusive process of article selection has lead us to have 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 specific area under study (when and what to automate testing). Beyond this field, we have no intention to generalize our results. Therefore, few problems with external validity are worthy of substantial attention.

8. Conclusions and future work

Automated software testing and development of test code are now mainstream in the software industry and challenging engineering topics on their own. Jeff Feldstein, who was a test manager at Cisco Systems, mentioned that "*We designed a test system that probably is as complicated as the system itself*". Yet, decision on when and what to automate in software testing has not been investigated with a literature review before. Given the importance of test automation and the large monetary investments that might be wasted with incorrect decisions we think such a study was needed.

This study makes four contributions. First, we found that the majority of prior works are opinion or experience papers that provide heuristics or guidelines while only a small share of papers present more rigorous work of evaluation or validation research presenting processes or models. This conflicts with the prior mapping studies of software engineering. Furthermore, we found that only 22% of the papers support their proposals with an empirical study of a real-world software system or project.

Second, we decided to include grey literature in our study as we found that the topic was of practical interest and as the academic studies of the topic were rare. To our knowledge, this paper is one the first MLRs (and SLRs) in software engineering that has studied grey literature in addition to the formally published literature. According to our experience in this MLR, we found that including grey literature in SLR studies is insightful, and thus the authors recommend including it when the topic has a low number of academic studies but high practitioner interest. The grey literature can help including practitioner experiences and opinions as part of SLRs.

Third, we qualitatively analyzed the factors affecting software test automation decisions. We found 15 factors that formed five groups: SUT-related factors, Test-related factors, Test-tool-related factors, Human and Organizational factors, and Cross-cutting and other factors. We found that the Need for regression testing (44 sources), Economic factors (43) and Maturity of SUT (39) are the most frequently mentioned factors. Finally, we note that the factor frequencies are not indication of factors' importance as the contextual variables of each case should determine the importance of each factor.

Fourth, we used our list of factors from Fig. 16 to create a checklist shown in Table 3 that can be used when making software automation decision. To our knowledge this checklist is the most comprehensive and the most scientific taxonomy and checklist so far.

We are planning to pursue the following future work directions:

- Classifying the decision-support approaches by the granularity of the decisions, e.g., in the test case level (which test cases should be automated?), in the project level (shall we use test automation at all?), or business level.
- Performing interview studies and surveys to understand the state of practice of software test automation decisions support. Particularly we are interested how do modern development concepts such Agile software development, DevOps and Continuous Deployment affect the decision-making. Even if one wishes to go for continuous delivery and fully automating tested there needs to prioritization order in automating the old tests.
- Validating our initial test automation checklist shown in Table 3 provides another avenue to future research.

Acknowledgements

Vahid Garousi was partially supported by several internal grants provided by the Hacettepe University and the Scientific and Technological Research Council of Turkey (TÜBİTAK) via grant #115E805. The authors would like to thank Professor Burak Turhan and Dr. Pilar Rodriguez from the University of Oulu and the anonymous reviewers for their comments on the earlier versions of this manuscript.

Pool of sources

Technical and scientific sources

[Source 1] Y. Amannejad, V. Garousi, R. Irving, and Z. Sahaf, "A Search-based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study," in Proc. of International Workshop on Regression Testing, co-located with the Sixth IEEE International Conference on Software Testing, Verification, and Validation, 2014, pp. 302–311.

[Source 2] C. Gouveia, J. Oliveira, and R. Quidute, "A way of Improving Test Automation Cost-Effectiveness," in Conference of the Association for Software Testing, 2006.

[Source 3] S. A. Jolly, V. Garousi, and M. M. Eskandar, "Automated Unit Testing of a SCADA Control Software: An Industrial Case Study based on Action Research," in IEEE International Conference on Software Testing, Verification and Validation (ICST), 2012, pp. 400–409.

[Source 4] S. K. Muthusundar, "Comparative study of test automaton ROI," Indian Journal of Computer Science and Engineering, vol. 2, 2011.

[Source 5] D. Hoffman, "Cost Benefits Analysis of Test Automation," in Software Testing Analysis & Review West conference (STARWEST), 1999.

[Source 6] A. Laapas, "Cost-benefit analysis of using test automation in the development of embedded software," School of Industrial Engineering and Management, Lappeenranta University of Technology, Finland, 2014.

[Source 7] R. Ramler and K. Wolfmaier, "Economic perspectives in test automation: balancing automated and manual testing with opportunity cost," in Proceedings of the 2006 international workshop on Automation of software test, 2006, pp. 85–91.

[Source 8] C. Persson and N. Yilmazturk, "Establishment of Automated Regression Testing at ABB: Industrial Experience Report on 'Avoiding the Pitfalls'," presented at the Proceedings of the IEEE international conference on Automated software engineering, 2004.

[Source 9] S. Berner, R. Weber, and R. K. Keller, "Observations and lessons learned from automated testing," in Proceedings of International Conference on Software Engineering, 2005, pp. 571–579.

[Source 10] U. Pranitha and B. V. Sastry, "Pragmatıc approach to software test automatıon," in Annual International Software Testing Conference (STC), 2013.

[Source 11] J. Kasurinen, O. Taipale, and K. Smolander, "Software test automation in practice: empirical observations," Adv. Soft. Eng., vol. 2010, pp. 1–13, 2010.

[Source 12] B. Boehmer and B. Patterson, "Software test automation–Developing an infrastructure designed for success," in STAREAST (Software Testing Analysis & Review East) Conference, 2001.

[Source 13] R. W. Rice, C. CSQA, and L. Rice Consulting Solutions, "Surviving the top 10 challenges of software test automation," CrossTalk: The Journal of Defense Software Engineering, pp. 26–29, 2003.

[Source 14] S. Münch, Peter Brandstetter, K. Clevermann, O. Kieckhoefel, and E. R. Schäfer, "The Return on Investment (ROI) of Test Automation," Pharmaceutical Engineering, vol. 32, pp. 1–8, 2012.

[Source 15] V. Motwani, "The When & How of Test Automation," in QAI India Ltd, 3rd Annual International Software Testing Conference, 2001.

[Source 16] E. Alégroth, R. Feldt, and L. Ryrholm, "Visual GUI testing in practice: challenges, problemsand limitations," Empirical Software Engineering, pp. 1–51, 2014/01/15 2014.

[Source 17] Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, and Y. Amannejad, "When to Automate Software Testing? Decision Support based on System Dynamics – An Industrial Case Study," in Proc. of International Conference on Software and Systems Process, 2014, pp. 149–158.

Books and book chapters

[Source 18] R. Black, Advanced Software Testing - Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager, 2nd ed.: Rocky Nook, 2014.

[Source 19] E. Dustin, J. Rashka, and J. Paul, Automated software testing: introduction, management, and performance: Addison–Wesley Professional, 1999.

[Source 20] J. A. Whittaker, J. Arbon, and J. Carollo, How Google Tests Software: Addison–Wesley Professional, 2012.

[Source 21] E. Dustin, T. Garrett, and B. Gauf, Implementing automated software testing: How to save time and lower costs while raising quality: Addison–Wesley Professional, 2009.

[Source 22] D. J. Mosley and B. A. Posey, Just Enough Software Test Automation: Prentice Hall Professional, 2002.

[Source 23] A. Clausen, "Project 1: Failure!, Project 2: Success!," in Experiences of Test Automation: Case Studies of Software Test Automation, D. Graham and M. Fewster, Eds., ed: Addison–Wesley Professional, 2012.

[Source 24] A. F. Benet, C. E. Lujua, H. S. Grau, M. M. Jáimez, F. M. Pérez, and C. Bianco, "Software for Medical Devices and Our Need for Good Software Test Automation," in Experiences of Test Automation: Case Studies of Software Test Automation, D. Graham and M. Fewster, Eds., ed: Addison–Wesley Professional, 2012.

[Source 25] M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools: Addison–Wesley, 1999.

[Source 26] S. Desikan and G. Ramesh, Software Testing: Principles and Practices: Pearson Education India, 2006.

Internet articles and white papers

[Source 27] M. Bartley, "Achieving business benefits through automated software testing," http://www.bcs.org/category/18128, 2012, Last accessed: May 2015.

[Source 28] S. Thompson, "Agile Testers, Should You Automate?," http://professionalservices.matrixresources.com/blog/agile-testers-should-you-automate, 2013, Last accessed: Jan. 2016.

[Source 29] Exforsys Inc., "Automated Testing Advantages, Disadvantages and Guidelines," http://www.exforsys.com/tutorials/testing/automated-testing-advantages-disadvantages-and-guidelines.html, 2005, Last accessed: Jan. 2016.

[Source 30] M. Clermont, "Automating tests vs. test-automation," http://googletesting.blogspot.com.tr/2007/10/automating-tests-vs-test-automation.html, 2007, Last accessed: Jan. 2016.

[Source 31] B. Galen, "Automation Selection Criteria – Picking the "Right" Candidates," http://www.compaid.com/caiinternet/ezine/Galen3.pdf, 2007, Last accessed: Jan. 2016.

[Source 32] S. Ford, "Automation Testing versus Manual Testing Guidelines," http://blogs.msdn.com/b/saraford/archive/2005/02/07/368833.aspx, 2005, Last accessed: Jan. 2016.

[Source 33] Cognizant, "Avoid Throwaway Test Automation" https://www.sqe.com/ControlImages/sqe/Image/Webinars/sqe_cognizant_finalslides.ppt, 2008, Last accessed: Jan. 2016.

[Source 34] B. L. Suer, "Choosing What To Automate," http://sqgne.org/presentations/2009-10/LeSuer-Jun-2010.pdf, 2010, Last accessed: Jan. 2016.

[Source 35] Galmont Consulting, "Determining What to Automate," http://galmont.com/wp-content/uploads/2013/11/Determining-What-to-Automate-2013_11.13.pdf, 2013, Last accessed: Jan. 2016.

[Source 36] L. Hayes, "Does Test Automation Save Time and Money?," http://www.stickyminds.com/article/does-test-automation-save-time-and-money, 2001, Last accessed: Jan. 2016.

[Source 37] I. Testy, "For those of you dreaming the 100% automation dream...please wake up!," http://blogs.msdn.com/b/imtesty/archive/2006/08/02/686010.aspx, 2006, Last accessed: Jan. 2016.

[Source 38] L. Hayes, "Functional Test Automation," in Testing SAP R/3: A Manager's Step-by-Step Guide, J. Fajardo and E. Dustin, Eds., ed: Wiley, 2007, Last accessed: Jan. 2016.

[Source 39] B. McLeod, "If you are going to run a test more than once, it should be automated.," http://www.teknologika.com/blog/tenet-if-you-are-going-to-run-a-test-more-than-once-it-should-be-automated/, 2005, Last accessed: Jan. 2016.

[Source 40] M. Larsen, "Learn When to Automate and When Not To: 99 Ways Workshop #5," http://www.mkltesthead.com/2013/07/99-ways-workshop-5-learn-when-to.html, 2013, Last accessed: Jan. 2016.

[Source 41] A. O'Doherty and Borland Software Corporation, "Manual vs. Automated Testing – explored," http://blog.borland.com/manualvsautomatedtesting/192/, 2013, Last accessed: Jan. 2016.

[Source 42] Gerrard Consulting, "Regression Testing – What to Automate and how," http://gerrardconsulting.com/sites/default/files/PaulGerrardRegressionTestingWhatAndHow.pptx, 1997, Last accessed: Jan. 2016.

[Source 43] Caritor Inc., "ROI on Test Automation - A Simple Yet Powerful Approach," http://www.cfoworld.co.uk/white-paper/software/4055/roi-on-test-automation–a-simple-yet-powerful-approach/, 2007, Last accessed: Jan. 2016.

[Source 44] R. Padmanaban, "Sources of Return on Investment (ROI) in Test Automation," http://www.qainfotech.com/blog/2012/07/sources-of-return-on-investment-roi-in-test-automation/, 2012, Last accessed: Jan. 2016.

[Source 45] B. Padilla, "Test Automation – Knowing When to Automate," http://oshyn.com/software-development/test-automation-when-automate, 2012, Last accessed: Jan. 2016.

[Source 46] Atlassian, "Test Automation and Best Practices," https://www.atlassian.com/test-automation, Last accessed: Jan. 2016.

[Source 47] Infosys, "Test Automation for Effective Post Deployment testing," http://www.infosys.com/IT-services/independent-validation-testing-services/white-papers/Documents/test-automation-post-deployment.pdf, 2007, Last accessed: Jan. 2016.

[Source 48] D. Graham, "Test Automation Patterns," http://www.dorothygraham.co.uk/patterns/desPatterns/index.html, Last accessed: Jan. 2016.

[Source 49] S. K. Gogineni and S. Kuchi, "Test Automation Process," http://stepauto.com/images/Santhi,%20Srihari-IBM%20pune.pdf, 2007, Last accessed: Jan. 2016.

[Source 50] I. M. Testy, "Test Automation ROI," http://blogs.msdn.com/b/imtesty/archive/2009/09/02/test-automation-roi-part-ii.aspx, 2009, Last accessed: Jan. 2016.

[Source 51] A. Narayanan and Aspire Systems, "Test Automation ROI Calculator," http://www.aspiresys.com/WhitePapers/whitepaper_Test_Automation_ROI_Calculator.pdf, 2010, Last accessed: Jan. 2016.

[Source 52] S. Carroll, "Test Automation: The Whens, Hows, and Whys," http://www.deriskit.com/news-article_test-automation-the-whens-hows-and-whys.html, 2014, Last accessed: Jan. 2016.

[Source 53] D. W. Johnson, "Test automation: When, how and how much," http://searchsoftwarequality.techtarget.com/tip/Test-automation-When-how-and-how-much, 2011, Last accessed: Jan. 2016.

[Source 54] AppLabs, "Test Automation-Delivering Business Value," http://www.informationweek.com/whitepaper/download/showPDF?articleID=6630000, 2008, Last accessed: Jan. 2016.

[Source 55] L. Crispin, "Testing in the Fast Lane: Automating Acceptance Testing in an Extreme Programming Environment," in Software Test Automation Conference, 2002, Last accessed: Jan. 2016.

[Source 56] E. Adams, "The Business Argument for Investing in Test Automation," http://www.ibm.com/developerworks/rational/library/content/RationalEdge/dec02/TestAutomation_TheRationalEdge_Dec2002.pdf, 2002, Last accessed: Jan. 2016.

[Source 57] M. Kelly, "The ROI of Test Automation," http://www.sqetraining.com/sites/default/files/articles/XDD8502filelistfilename1_0.pdf, 2007, Last accessed: May 2015.

[Source 58] K. Stobie, "Too much automation or not enough? When to automate," http://www.uploads.pnsqc.org/2009/papers/Keith_Stobie_Too_Much_Automation_paper.pdf, 2009, Last accessed: Jan. 2016.

[Source 59] S. Rowe, "Too Much Test Automation?," http://blogs.msdn.com/b/steverowe/archive/2005/01/28/362668.aspx, 2005, Last accessed: Jan. 2016.

[Source 60] Various member of the StackExchange community, "What are good guidelines to determine when to automate a test," http://sqa.stackexchange.com/questions/822/what-are-good-guidelines-to-determine-when-to-automate-a-test, 2012, Last accessed: Jan. 2016.

[Source 61] mVerify Corporation, "What's my Testing ROI?," http://robertvbinder.com/wp-content/uploads/sites/4/2011/06/Whats-My-Testing-ROI1.pdf, 2011, Last accessed: Jan. 2016.

[Source 62] E. J. Correia, "What's Your Automation Index?," http://www.testpoint.com.au/attachments/093_What's%20Your%20Automation%20Index.pdf, 2007, Last accessed: Jan. 2016.

[Source 63] BQurious Software Inc., "When and How Much to Automate," http://www.bqurious.com/when-and-how-much-to-automate, 2014, Last accessed: Jan. 2016.

[Source 64] O. Laursen, "When automated tests make sense, and when they don't," http://ole-laursen.blogspot.com.tr/2012/03/when-automated-tests-make-sense-and.html, 2012, Last accessed: Jan. 2016.

[Source 65] B. Marick, "When Should a Test Be Automated?," http://www.exampler.com/testing-com/writings.html, 1998, Last accessed: Jan. 2016.

[Source 66] A. M. employee, "When To Automate," http://blogs.msdn.com/b/micahel/archive/2005/12/14/whentoautomate.aspx, 2005, Last accessed: Jan. 2016.

[Source 67] D. Weiss, "When to Automate Testing," http://davidweiss.blogspot.com.tr/2006/08/when-to-automate-testing.html, 2006, Last accessed: Jan. 2016.

[Source 68] J. Fernandes and A. D. Fonzo, "When to Automate Your Testing (and When Not To)," http://www.oracle.com/technetwork/cn/articles/when-to-automate-testing-1-130330.pdf, 2013, Last accessed: Jan. 2016.

[Source 69] TestingWhiz, "When to do Manual Testing and when to Automated Testing," http://blog.testing-whiz.com/2011/12/when-to-do-manual-testing-and-when-to.html, 2011, Last accessed: Jan. 2016.

[Source 70] S. Rowe, "When to Test Manually and When to Automate," http://blogs.msdn.com/b/steverowe/archive/2008/02/26/when-to-test-manually-and-when-to-automate.aspx, 2008, Last accessed: Jan. 2016.

[Source 71] Intense Testing Group, "Why and When to Perform Automation Testing," http://intensetesting.wordpress.com/2013/11/07/why-and-when-to-perform-automation-testing/, 2013, Last accessed: Jan. 2016.

[Source 72] Software Testing Class, "Why How and When to Automate Testing," http://www.softwaretestingclass.com/why-how-and-when-to-automate-software-testing/, 2014, Last accessed: Jan. 2016.

YouTube videos

[Source 73] D. Graham, "Intelligent Mistakes in Test Automation," https://www.youtube.com/watch?v=Dbim6ZkSHBg, 2013, Last accessed: Jan. 2016.

[Source 74] Rice Consulting Services Inc., "What to Automate First," http://www.youtube.com/watch?v=eo66ouKGyVk, 2014, Last accessed: Jan. 2016.

Tools

[Source 75] IBM Rational, "IBM Rational Quality Management ROI Calculator," in http://www-01.ibm.com/software/rational/offerings/testing/roi/tool/ROI_Rational.html, Last accessed: Jan. 2016.

[Source 76] Elbrus Ltd., "Return on Investment Calculator for Test Automation," in http://www.elbrus.com/services/test_automation_roi_calc/, Last accessed: Jan. 2016.

[Source 77] GlobalNow Inc., "ROI Calculator by GlobalNowIT," in http://globalnowit.com/calculator/, Last accessed: Jan. 2016.

[Source 78] Automated Testing Institute, "Test Automation ROI Calculator," in http://www.automatedtestinginstitute.com/home/index.php?Itemid=65, Last accessed: Jan. 2016.

References

[1] HP Capgemini. Sogetti, World quality report 2014-2015. www.sogeti.com/solutions/testing/wqr/, Last accessed: Sept. 2015.

[2] T. Britton, L. Jeng, G. Carver, P. Cheak, and T. Katzenellenbogen, "Reversible debugging software," University of Cambridge, Judge Business School, *Tehnical Report*, 2013.

[3] O. Taipale, J. Kasurinen, K. Karhu, K. Smolander, Trade-off between automated and manual software testing, Int. J. Syst. Assurance Eng. Manage. 2 (2011) 114–125.

[4] D. Huizinga, A. Kolawa, Automated Defect Prevention: Best Practices in Software Management, Wiley-IEEE Computer Society Press, 2007.
[5] J. Bach, Test automation snake oil, in: Proceedings of International Conference and Exposition on Testing Computer, 1997.
[6] D.M. Rafi, K.R.K. Moses, K. Petersen, M.V. Mantyla, Benefits and limitations of automated software testing- systematic literature review and practitioner survey, in: International Workshop on Automation of Software Test, 2012, pp. 36–42.
[7] D. Hoffman, Cost-benefits analysis of test automation, Software Testing Analysis and Review Conference (STARWEST), 1999.
[8] V. Garousi, J. Zhi, A survey of software testing practices in Canada, J. Syst. Softw. 86 (2013) 1354–1376.
[9] V. Garousi, A. Coşkunçay, A.B. Can, O. Demirörs, A survey of software engineering practices in Turkey, J. Syst. Softw. 108 (2015) 148–177.
[10] K. Stobi, Too much automation or not enough? When to automate testing, Pacific Northwest Software Quality Conference, 2009.
[11] D.J. Mosley, B.A. Posey, Just Enough Software Test Automation, Prentice Hall Professional, 2002.
[12] C. Kaner, J. Bach, B. Pettichord, Lessons Learned in Software Testing, John Wiley & Sons Inc, 2001.
[13] V. Garousi, A systematic approach to software test automation and how to increase its ROI, Invited Talk, TestIstanbul industry conference, 2013.
[14] V. Garousi, Recent trends in software testing: opportunities for industry-academia collaborations, Invited speaker, YouTube Corporation, 2010 June 30.
[15] V. Garousi, K. Herkiloğlu, Selecting the right topics for industry-academia collaborations in software testing: an experience report, IEEE International Conference on Software Testing, Verification, and Validatio, 2016.
[16] Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, Y. Amannejad, When to automate software testing? Decision support based on system dynamics – an industrial case study, in: Proc. of International Conference on Software and Systems Process, 2014, pp. 149–158.
[17] K.C. Archie, O.R. Fonorow, M.C. McGould, R.E. McLear, E.C. Read, E.M. Schaefer, et al., "Test automation system," ed: US Patent #US5021997, 1991.
[18] V. Garousi, R. Kotchorek, M. Smith, Test cost-effectiveness and defect density: A case study on the android platform, Adv. Comput. 89 (2013) 163–206.
[19] J. Feldstein, How to recruit, motivate, and energize superior test, Dec. 2014, Last accessed: http://www.youtube.com/watch?v=PyhtoQz7RHY.
[20] G. Meszaros, xUnit Test Patterns, Pearson Education, 2007 http://xunitpatterns.com.
[21] A. Page, K. Johnston, How We Test Software at Microsoft, Microsoft Press, 2008.
[22] J.A. Whittaker, J. Arbon, J. Carollo, How Google Tests Software, Addison-Wesley Professional, 2012.
[23] P. Ammann, J. Offutt, Introduction to Software Testing, Cambridge University Press, 2008.
[24] K. Stobie, "Too much automation or not enough? When to automate testing," 2009.
[25] S. Desikan, G. Ramesh, Software Testing: Principles and Practices, Pearson Education India, 2006.
[26] E. Dustin, T. Garrett, B. Gauf, Implementing automated software testing: How to save time and lower costs while raising quality, Addison-Wesley Professional, 2009.
[27] A. Ampatzoglou, A. Ampatzoglou, A. Chatzigeorgiou, P. Avgeriou, The financial aspect of managing technical debt: A systematic literature review, Inform. Softw. Technol. 64 (2015) 52–73 8//.
[28] R.L. Glass and T. DeMarco, *Software Creativity 2.0*: developer.* Books, 2006.
[29] R.T. Ogawa, B. Malen, Towards rigor in reviews of multivocal literatures: applying the exploratory case study method, Revi. Edu. Res. 61 (1991) 265–286.
[30] W.F. Whyte, Participatory Action Research, SAGE Publications, 1990.
[31] K.M. Benzies, S. Premji, K.A. Hayden, K. Serrett, State-of-the-evidence reviews: advantages and challenges of including grey literature, Worldv. Evidence-Based Nursing 3 (2006) 55–61.
[32] Q. Mahood, D. Van Eerd, E. Irvin, Searching for grey literature for systematic reviews: challenges and benefits, Res. Syn. Methods 5 (2014) 221–234.
[33] S. Hopewell, M. Clarke, S. Mallett, Grey literature and systematic reviews, in: H.R. Rothstein, A.J. Sutton, M. Borenstein (Eds.), Publication Bias in Meta-Analysis: Prevention, Assessment and Adjustments, John Wiley & Sons, 2006.
[34] H. S, M. S, C. M, E. M, Grey literature in meta-analyses of randomized trials of health care interventions, Cochrane Datab. Syst. Rev. (2007).
[35] E. Tom, A. Aurum, R. Vidgen, An exploration of technical debt, J. Syst. Softw. 86 (2013) 1498–1516.
[36] I. Kulesovs, "iOS Applications Testing," vol. 3, pp. 138–150, 2015.
[37] M. Sulayman, E. Mendes, A systematic literature review of software process improvement in small and medium web companies, in: D. Ślęzak, T.-h. Kim, A. Kiumi, T. Jiang, J. Verner, S. Abrahão (Eds.), Advances in Software Engineering, 59, Springer Berlin Heidelberg , 2009, pp. 1–8.
[38] A. Yasin, M.I. Hasnain, On the quality of grey literature and its use in information synthesis during systematic literature reviews Master Thesis, Blekinge Institute of Technology, Sweden, 2012.
[39] V. Garousi, M. Felderer, M.V. Mäntylä, The need for (more) multivocal literature reviews in software engineering, Under review, International Conference on Evaluation and Assessment in Software Engineering (EASE) *paper PDF:*, 2016 https://goo.gl/O0JPsF .
[40] V. Garousi, Online Paper Repository for Systematic Mapping of Secondary studies in software testing, http://goo.gl/Oxb0x8, Last accessed: Sept. 2015.
[41] W. Afzal, R. Torkar, 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.
[42] P.A.d.M.S. Neto, I.d.C. Machado, J.D. McGregord, E.S.d. Almeida, S.R.d.L. Meira, A systematic mapping study of software product lines testing, Inform. Softw. Technol. 53 (2011) 407–423.
[43] I. Banerjee, B. Nguyen, V. Garousi, A. Memon, Graphical user interface (GUI) testing: systematic mapping and repository, Inform. Softw. Technol. 55 (2013) 1679–1694.
[44] Ç. Çatal, D. Mishra, Test case prioritization: a systematic mapping study, Softw. Quality J. 21 (2013) 445–478.
[45] V.G. Yusifoğlu, Y. Amannejad, A. Betin-Can, Software test-code engineering: a systematic mapping, J. Inform. Softw. Technol. (2014).
[46] A.C.D. Neto, R. Subramanyan, M. Vieira, 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.
[47] B. Haugset, G.K. Hanssen, Automated Acceptance testing-a literature review and an industrial case study, in: Agile Conference, 2008, pp. 27–38.
[48] P.K. Singh, O.P. Sangwan, 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.
[49] S. Doğan, A. Betin-Can, V. Garousi, Web application testing: a systematic literature reviewr, J. Syst. Softw. 91 (2014) 174–201.
[50] U. Kanewala, J.M. Bieman, Testing scientific software: a systematic literature review, Inform. Softw. Technol. 56 (2014) 1219–1232 10//.
[51] D. Lee, M. Yannakakis, Principles and methods of testing finite state machines-a survey, in: Proceedings of the IEEE, 84, 1996, pp. 1090–1123.
[52] S. Yoo, M. Harman, Regression testing minimization, selection and prioritization: a survey, Softw. Test. Verif. Reliab. 22 (2012) 67–120.
[53] M. Bozkurt, M. Harman, Y. Hassoun, Testing and verification in service-oriented architecture-a Survey, Softw. Test. Verif. Reliab. 23 (2013) 261–313.
[54] M. Shirole, R. Kumar, UML behavioral model based test case generation: a survey, ACM SIGSOFT Softw. Eng. Notes 38 (2013) 1–13.
[55] M. Harman, P. McMinn, M. Shahbaz, S. Yoo, A comprehensive survey of trends in oracles for software testing, IEEE Trans. Softw. Eng. (2014).
[56] B. Kitchenham, P. Brereton, 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.
[57] 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.
[58] M. Bell, P. Cordingley C. Isham, 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.
[59] H. Aveyard, Doing A Literature Review In Health And Social Care: A Practical Guide: A Practical Guide Paperback, 2 edition, Open University Press, 2010.
[60] V. Garousi, A. Mesbah, A. Betin-Can, S. Mirshokraie, A systematic mapping of web application testing, Inform. Softw. Technol. (2013).
[61] I. Banerjee, B. Nguyen, V. Garousi, A. Memon, Graphical user interface (GUI) testing: systematic mapping and repository, Inform. Softw. Technol. (2013).
[62] N. Bencomo, S. Hallsteinsen, E. Santana de Almeida, A View of the Dynamic Software Product Line Landscape, IEEE Comput. 45 (2012) 36–41.
[63] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 71–80.
[64] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Inform. Softw. Technol. 64 (2015) 1–18.
[65] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, presented at the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE), 2008.
[66] B. Kitchenham, S. Charters, Guidelines for Performing systematic literature reviews in software engineering," in evidence-based software engineering, Evidence-Based Softw. Eng. (2007).
[67] S. Ali, L.C. Briand, H. Hemmati, R.K. Panesar-Walawege, A systematic review of the application and empirical investigation of search-based test-case generation, IEEE Trans. Softw. Eng. 36 (2010) 742–762.
[68] F. Elberzhager, J. Münch, V.T.N. Nha, A systematic mapping study on the combination of static and dynamic quality assurance techniques, Inform. Softw. Technol. 54 (2012) 1–15.
[69] V. Garousi, Classification and trend analysis of UML books (1997-2009), J. Softw. Syst. Model. (SoSyM) (2011).
[70] V. Garousi, Y. Amannejad, A. Betin-Can, Software test-code engineering: a systematic mapping, J.f Inform. Softw. Technol. 58 (2015) 123–147.
[71] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empirical Softw. Eng. 16 (2011) 365–395.

[72] M. Fewster, Common mistakes in test automation, http://www.agileconnection.com/sites/default/files/article/file/2012/XDD2901filelistfilename1_0.pdf, 2012. Last accessed: Sept. 2015.
[73] K. Karhu, T. Repo, O. Taipale, K. Smolander, Empirical observations on software testing automation, in: Software Testing Verification and Validation, 2009. ICST '09. International Conference on, 2009, pp. 201–209.
[74] B. Pettichord, Seven steps to test automation success, (2001), Last accessed: Sept. 2015 https://www.prismnet.com/~wazmo/papers/seven_steps.
[75] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: presented at the Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, United Kingdom, 2014.
[76] V. Garousi, M. V. Mäntylä, Online paper repository for the mlr on 'when and what to automate in software testing?' (2016), Last accessed: Jan http://goo.gl/zwY1sj
[77] D.S. Cruzes and T. Dybå, "Synthesizing evidence in software engineering research," in *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010
[78] D.S. Cruzes, T. Dybå, Recommended steps for thematic synthesis in software engineering, in: Proc. International Symposium on Empirical Software Engineering and Measurement, 2011, pp. 275–284.
[79] D.S. Cruzesa, T. Dybåb, Research synthesis in software engineering: a tertiary study, Inform. Softw. Technol. 53 (2011) 440–455.
[80] G.S. Waliaa, J.C. Carverb, A systematic literature review to identify and classify software requirement errors, Inform. Softw. Technol. 51 (2009) 1087–1109.
[81] S. Ali, L.C. Briand, H. Hemmati, R.K. Panesar-Walawege, A systematic review of the application and empirical investigation of searchbased test case generation, IEEE Trans. Softw. Eng. 36 (2010) 742–762.
[82] H. Cooper, L.V. Hedges, J.C. Valentine, The Handbook of Research Synthesis and Meta-Analysis, 2nd ed, Russell Sage Foundation, 2009.
[83] M.B. Miles, A.M. Huberman, J. Saldana, Qualitative Data Analysis: A Methods Sourcebook, Third Edition, SAGE Publications Inc, 2014.
[84] B. Daniel, V. Jagannath, D. Dig, D. Marinov, ReAssert: suggesting repairs for broken unit tests, in: presented at the Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, 2009.
[85] Y. Amannejad, V. Garousi, R. Irving, Z. Sahaf, A search-based approach for cost-effective software test automation decision support and an industrial case study, in: Proc. of International Workshop on Regression Testing, co-located with the Sixth IEEE International Conference on Software Testing, Verification, and Validation, 2014, pp. 302–311.
[86] M. Kelly, The ROI of test automation, http://www.sqetraining.com/sites/default/files/articles/XDD8502filelistfilename1_0.pdf, 2007. Last accessed: May 2015.
[87] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, 2000.
[88] E. Arisholm, H. Gallis, T. Dyba, D.I.K. Sjoberg, Evaluating pair programming with respect to system complexity and programmer expertise, IEEE Trans. Softw. Eng. 33 (2007) 65–86.