Information and Software Technology 80 (2016) 195–216

Contents lists available at ScienceDirect

Information and Software Technology

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

A systematic literature review of literature reviews in software testing

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 17 December 2015
Revised 8 September 2016
Accepted 9 September 2016
Available online 12 September 2016

Keywords:
Secondary studies
Tertiary study
Software testing
Systematic mapping
Systematic literature reviews
Surveys

ABSTRACT

Context: Any newcomer or industrial practitioner is likely to experience difficulties in digesting large volumes of knowledge in software testing. In an ideal world, all knowledge used in industry, education and research should be based on high-quality evidence. Since no decision should be made based on a single study, secondary studies become essential in presenting the evidence. According to our search, over 101 secondary studies have been published in the area of software testing since 1994. With this high number of secondary studies, it is important to conduct a review in this area to provide an overview of the research landscape in this area.

Objective: The goal of this study is to systematically map (classify) the secondary studies in software testing. We propose that tertiary studies can serve as summarizing indexes which facilitate finding the most relevant information from secondary studies and thus supporting evidence-based decision making in any given area of software engineering. Our research questions (RQs) investigate: (1) Software-testing-specific areas, (2) Types of RQs investigated, (3) Numbers and Trends, and (4) Citations of the secondary studies.

Method: To conduct the tertiary study, we use the systematic-mapping approach. Additionally, we contrast the testing topics to the number of Google hits to address a general popularity of a testing topic and study the most popular papers in terms of citations. We furthermore demonstrate the practicality and usefulness of our results by mapping them to ISTQB foundation syllabus and to SWEBOK to provide implications for practitioners, testing educators, and researchers.

Results: After a systematic search and voting process, our study pool included 101 secondary studies in the area of software testing between 1994 and 2015. Among our results are the following: (1) In terms of number of secondary studies, model-based approach is the most popular testing method, web services are the most popular system under test (SUT), while regression testing is the most popular testing phase; (2) The quality of secondary studies, as measured by a criteria set established in the community, is slowly increasing as the years go by; and (3) Analysis of research questions, raised and studied in the pool of secondary studies, showed that there is a lack of 'causality' and 'relationship' type of research questions, a situation which needs to be improved if we, as a community, want to advance as a scientific field. (4) Among secondary studies, we found that regular surveys receive significantly more citations than SMs ($p = 0.009$) and SLRs ($p = 0.014$).

Conclusion: Despite the large number of secondary studies, we found that many important areas of software testing currently lack secondary studies, e.g., test management, role of product risk in testing, human factors in software testing, beta-testing (A/B-testing), exploratory testing, testability, test stopping criteria, and test-environment development. Having secondary studies in those areas is important for satisfying industrial and educational needs in software testing. On the other hand, education material of ISTQB foundation syllabus and SWEBOK could benefit from the inclusion of the latest research topics, namely search-based testing, use of cloud-computing for testing and symbolic execution.

Contents

* Corresponding author at: Software Engineering Research Group, Department of Computer Engineering, Hacettepe University, Ankara, Turkey.
E-mail addresses: vgarousi@gmail.com, vahid.garousi@hacettepe.edu.tr (V. Garousi), mika.mantyla@oulu.fi (M.V. Mäntylä).

http://dx.doi.org/10.1016/j.infsof.2016.09.002
0950-5849/

1. Introduction

Secondary studies are common in software engineering (SE). A secondary study is defined as a study of studies [1], i.e., a review of individual (or, primary) studies. Example types of secondary studies include: regular surveys, Systematic Literature Reviews (SLR), and Systematic Mapping (SM) studies.

Software testing is an active area of SE. According to our search, over 101 secondary studies have been published in the area of software testing since 1994. With this high number of secondary studies in this area, it is important to conduct a tertiary review in this area to provide an overview of the research landscape in this area. A tertiary review is a study of secondary studies (or, a systematic review of systematic reviews) [2]. Tertiary studies 'review the reviews' in a given area in order to provide an overview of the state of evidence in that area. The SE community as a whole believes that secondary and tertiary studies are useful, e.g., [2–5]. There are relatively high number of citations to secondary and tertiary studies in SE, and also there are studies such as [4] which report the usefulness and value of these studies.

There have been tertiary studies in various areas of SE (e.g., [6–15]), but none focusing on testing yet. As discussed above, any newcomer researcher or industrial practitioner is likely to experience difficulties in digesting large volumes of knowledge in software testing. Also, in an ideal world, all knowledge used in industry, education and research should be based on high-quality evidence. Since no decision should be made based on a single study, secondary studies become essential in presenting the evidence. We propose that tertiary studies can serve as summarizing indexes which would facilitate it to find the most relevant information from secondary studies and thus supporting evidence-based decision making in any given area of software engineering.

The authors believe that a tertiary studies should be like the "index" of a book. Such a tertiary study will be useful in that it is read first by the people (e.g., new PhD students and practitioners) who want to know what is out there in a given area (software testing, in our case). A complain often heard from practitioners is that academic literature is impenetrable due to the sheer volume of the literature. A tertiary study such as the current paper should make it more penetrable. Also, such a tertiary study could be used as an aid when constructing contents of research-intensive courses on software testing. If a sub-field of software testing has a large body of research literature, then a review of this literature (via secondary studies) has most likely been performed or should be performed.

Based on the above needs and motivations, in this work, we systematically classify the body of knowledge in secondary studies in software testing via a tertiary study [16]. Our study aims at answering the following four research questions (RQs):

- RQ1: What software-testing-specific areas have been investigated in the secondary studies? Answering this RQ will enable us to determine the software-testing-specific areas covered and not covered by secondary studies. Knowing the areas not covered will pinpoint the need for conducting secondary studies in those areas.
- RQ2: What types of RQs are being investigated? This allows us to characterize the studies in software testing from the viewpoint of philosophy of science. This can help us find gaps and trends in type of secondary studies being conducted.
- RQ3: What are the annual trends of types, quality, and number of primary studies reviewed by the secondary studies? Answering this RQ will allow us to get a big picture of the landscape in this area.
- RQ4: What are the highest cited secondary studies and are the secondary studies cited more often than primary studies? Given the importance of citations to determine scientific merit, we decided to investigate what secondary studies are the most cited. For the same reason, we investigate whether secondary studies receive more citations than primary studies.

As s part of this study, we define inclusion (selection) and exclusion criteria of relevant secondary studies, and systematically develop and refine a systematic map (classification schema) of all the selected studies.

The remainder of study is organized as follows. Section 2 reviews the related work. Section 3 describes our research method, including the overall SM process, the goal and research questions tackled in this study. Section 4 discusses the article selec-

Table 1
List of tertiary studies in SE (sorted by year of publication).

#	Topic	Number of secondary studies	Year	Ref.
1	SLRs in SE – A SLR	20	2009	[6]
2	SLRs in SE– A tertiary study	33	2010	[7]
3	Critical appraisal of SLRs in SE from the perspective of the research questions	53	2010	[8]
4	Research synthesis in SE-A tertiary study	49	2011	[9]
5	Six years of SLRs in SE-An updated tertiary study	67	2011	[10]
6	Signs of Agile Trends in Global SE Research-A Tertiary Study	12	2011	[11]
7	Systematic approach for identifying relevant studies in SE	38	2011	[12]
8	SLRs in Distributed SE-A Tertiary Study	14	2012	[13]
9	A tertiary study: experiences of conducting SLRs in SE	116	2013	[14]
10	Risks and risk mitigation in global software development: A tertiary study	37	2014	[15]

tion process. Section 5 presents the systematic map which has been built through an iterative selection and synthesis process. Section 6 presents the results of the tertiary study based on systematic mapping. Section 7 summarizes the findings, points out the implications of the results, and discusses the potential threats to validity of our study. Finally, Section 8 concludes this study and states the future work directions.

2. Background and related work

As the background and related work, we review the other tertiary studies in SE, and the research questions investigated in those tertiary studies.

2.1. Other tertiary studies in SE

This work is a tertiary study in the area of software testing, in which the secondary studies in software testing are studied. As stated by Kitchenham and Charters [2], a tertiary review is a systematic review of systematic reviews, also called a 'meta-systematic-review'.

We did not find any tertiary study in software testing. However, there have been a handful number of tertiary studies in SE. We briefly review them next because of research setting similarities among those works and ours. Our search for tertiary studies in SE returned 10 results [6–15] which have been listed in Table 1, sorted by their years of publication. We have also specified the number of secondary studies reviewed by each tertiary study and its year of publication.

We can see that tertiary studies in SE have started to appear since recently, i.e., after 2009. The number of secondary studies reviewed by each tertiary study ranges from 12 to 116. Seven of the tertiary studies do not focus on specific sub-areas of SE but were rather on the general SE, while two [11,13] have focused on specific sub-areas in SE: trends of Agile practices in the global/distributed SE [11], and SLRs in global/distributed SE [13].

The work of Kitchenham et al. [6] in 2009 seems to be the first tertiary study in SE, which was updated a year later "*This study [7] updates our previous study [6] using a broad automated search*". They found an additional 35 SLRs corresponding to 33 unique studies. Of these papers, 17 appeared relevant to the undergraduate educational curriculum and 12 appeared of possible interest to practitioners. The authors reported that the number of SLRs being published was increasing. The quality of papers in conferences and workshops has improved as more researchers were using SLR guidelines. Another update was reported in [10]. The goal was to extend and update the two previous tertiary studies [6,7] to cover the period between July 2008 and December 2009. The authors analyzed the quality, coverage of SE topics, and potential impact of published SLRs for education and practice. The findings suggested that the SE research community is starting to adopt SLRs consistently as a research method.

Silva and Santos conducted a critical appraisal of SLRs in SE from the perspective of the research questions asked in the reviews [8]. They analyzed 53 SLRs that had been collected in two earlier published tertiary studies [6,7]. The study found that over 65% of the research questions asked in the reviews were exploratory and only 15% investigated causality questions. The authors concluded that there is a need for a consistent use of terminology to classify secondary studies and that reports of literature reviews should follow reporting guidelines to support assessment and comparison.

The objective of [9] was to contribute to a better understanding of the types, methods, and challenges in synthesizing SE research and their implications for the progress of research and practice. Among the results were the following. As many as half of the 49 reviews included in that study did not contain any type of formal synthesis.

The tertiary study reported in [11] aimed at detecting "signs" of Agile trends in global/distributed SE research. The study reported that, in contrast to recent beliefs that agile and distributed are two incompatibilities, global agile development has become more and more accepted. The study concluded that there are indications that both globalization and "agilization" of software companies are stable trends for the future but that there is a need for further studies on the topic.

The guideline paper [12] provides a systematic approach for searching for and identifying relevant studies in SE. The authors argue that one critical step in conducting SM and SLR studies is to design and execute appropriate and effective search strategy. This is a time-consuming and error-prone step, which needs to be carefully planned and implemented. There is an apparent need for a systematic approach to designing, executing, and evaluating a suitable search strategy for optimally retrieving the target literature from digital libraries.

The tertiary study reported in [13] analyzed the SLRs in the area of Distributed Software Development (DSD). Of 14 SLRs in the pool of studies, seven addressed aspects of managing distributed development. Four SLRs addressed topics of engineering process. The three remaining were related to requirements, design and SE education in the context of DSD.

Experiences of conducting SLRs in SE were discussed in the form of a tertiary study in [14]. The authors gathered the experiences published by researchers in 116 SLRs. Findings highlighted that search strategy, online databases, planning and data extraction are the most challenging phases of a SLR. Lack of standard terminology in SE papers, poor quality of abstracts and problems with search engines are some of the most cited challenges. The study [15] was a tertiary study reporting the risks and risk mitigation in global software development.

Additionally, an online resource named SE evidence map, as part of the Evidence-Based SE (EBSE) website, have been published as an online resource [3] which provides a simple classification of 22 SLR studies.

Table 2
RQs raised in the existing SE tertiary studies.

Ref.	Tertiary study topic	RQs
[6]	SLRs in SE: A SLR	• RQ1: How much SLR activity has there been since 2004? • RQ2: What research topics are being addressed? • RQ3: Who is leading SLR research? • RQ4: What are the limitations of current research?
[7]	SLRs in SE: A tertiary study	• RQ1: How many SLRs were published between 2004 and 2008? • RQ2: What research topics are being addressed? • RQ3: Which individuals and organizations are most active in SLR: based research? • RQ4: Are the limitations of SLRs still an issue? • RQ5: Is the quality of SLRs improving?
[8]	A critical appraisal of SLRs in SE from the perspective of the research questions	• RQ1: What are the types of research questions asked in literature reviews? • RQ2: Are the research questions explicitly presented in the studies? • RQ3: How the research questions are used to guide the search of primary studies? • RQ4: Is the type of research questions used in the studies related to the classification of mapping studies and systematic reviews?
[9]	Research synthesis in SE: A tertiary study	• RQ1: What is the basis in terms of primary study types and evidence that is included? • RQ2: How, and according to which methods, are the findings of systematic reviews in SE synthesized? • RQ3: How are the syntheses of the findings presented?
[10]	Six years of SLRs in SE: An updated tertiary study (follow: up to [6] and [7])	• RQ1: How many SLRs were published between 1st January 2004 and 31st December 2009? • RQ2: What research topics are being addressed? • RQ3: Which individuals and organizations are most active in SLR: based research? • RQ4: Are the limitations of SLRs, as observed in the two previous studies, still an issue? • RQ5: Is the quality of SLRs improving?
[13]	Distributed SE	• RQ1: How many systematic literature reviews have been published in the Distributed Software Development (DSD) context? • RQ2: What research topics are being addressed? • RQ3: What research questions are being investigated? • RQ4: Which individuals and organizations are involved in SLR-based DSD research? • RQ5: What are the limitations of systematic literature reviews in DSD?

On another perspective, the authors of this article has had recent experience in conducting and reporting several SM and SLR studies in SE. Several of those studies have been in software testing [17–22], while several others, e.g., [23–27], focused on other areas of SE, i.e., development of scientific software [23], UML-driven software performance engineering [25], software documentation [28], UML books [24], pair-programming [27], usage of metrics in Agile software development [29]. The experience and expertise gained in conducting those SM and SLR studies have been influential in conducting this tertiary study.

2.2. A review of research questions investigated in other SE tertiary studies

As discussed in Section 2, we found ten tertiary studies in SE [6–15], see Table 1. Out of those nine tertiary studies, we selected six tertiary studies [6–10,13] whose contexts and RQs were relevant to our topic. The remaining four tertiary studies [11,12,14,15] were focused on topics not that relevant to our study, and thus, we did not use their RQs as a baseline for deriving the RQs of this study. They were:

- Signs of Agile trends in global SE research: a tertiary study [11]
- A tertiary study: experiences of conducting SLRs in SE [14]
- Systematic approach for identifying relevant studies in SE [12]
- Risks and risk mitigation in global software development: A tertiary study [15]

We used the RQs in Table 2, to guide our own selection of research questions.

3. Research method

A tertiary review uses the same methodology as a standard SLR [2]. For conducting our tertiary study, we had the choice of using the SM or the SLR approach. Similar to other research fields, SE has its methodologies for conducting secondary studies. Petersen et al. [16] presented a guideline study on how to conduct SM studies in SE. The guideline study by Petersen et al. [16] provided us with insights on building classification schemes and structuring a particular sub-domain of interest in SE, which we have followed in our previous and recent SM and SLR studies [17–20,23–26].

Kitchenham et al. also presented in [2] detailed guidelines for performing SLR studies in SE, most of which could also be used for a SM study. The guidelines described by Petersen et al. [16,30] and also Kitchenham et al. [2] were followed in our SM study. Justification to follow the guidelines from those two studies is that they are treated as two comprehensive guidelines to conduct SLR and SM in SE and have been used by many other researchers conducting and reporting SLRs and SMs, e.g., [24,31,32]. In designing the methodology for this SM, methods from several other SMs such as [24,31,32] were also incorporated. In the following, the goal and research questions of our study are presented. Then, an overview of our research process is discussed.

3.1. Goal and research questions

The research approach we have used in our study is the Goal-Question-Metric (GQM) methodology [33]. Based on the GQM goal template, the goal of this study is to systematically map (classify) the state-of-the-art in secondary studies in the area of software testing, to find out the recent trends and directions in this field and to identify opportunities for future research from the point of view of researchers and practitioners in the context of research literature in this area. We also want to understand how the research space has evolved over time with regards to the above research attributes.

Using the GQM approach [33], similar to regular SLR and also tertiary studies in SE [6–15], we intended to raise a set of research questions (RQs) based on the above research goal. To ensure the novelty, usefulness and relevance of our RQs, we reviewed in Section 2.2 the RQs raised in other tertiary studies in SE. Based on our study's goal, and also keeping the RQs raised in other tertiary studies, we raise and investigate four main RQs in this study:

- RQ1: What software-testing-specific areas have been investigated in the secondary studies?
  Answering this RQ will enable us to determine the software-testing-specific areas covered and not covered by secondary studies. Knowing the areas not covered will pinpoint the need for conducting secondary studies in those areas. To classify the software-testing-specific questions, we covered the dimensions of the testing method, the type of system under test (SUT), and the testing phase. More details about these software-testing-specific areas are discussed when we present our study's systematic map (Section 5.2).
- RQ2: What types of RQs are being investigated?
  We answered this RQ by using RQ classification presented by Easterbrook et al. [34], e.g., exploratory RQs, base-rate RQs, relationship RQs and causality RQs. This allows us to characterize the studies in software testing from the viewpoint of philosophy of science. This can help us find gaps and trends in type of secondary studies being conducted.
- RQ3: What are the annual trends of the secondary studies?
  This RQ presents the trends and demographic data typically reported in SMs and SLRs with respect to number of secondary studies, number of papers included in secondary studies papers, and types of secondary studies (i.e., surveys, SMs, SLRs). Note that the survey-type papers that we consider as secondary studies are not those studies which collect data from respondents by questionnaires and then present the analyzed results. But instead, we consider papers which have surveyed the papers in a given area of testing (e.g., model-based testing) and presented a classification and/or summary of those studies, e.g., a paper entitled "*A survey on model-based testing approaches*".
- RQ4: What are the highest cited secondary studies and are the secondary studies cited more often than primary studies?
  Given the importance of citations to determine scientific merit, we decided to investigate what secondary studies are the most cited. For the same reason, we investigate whether secondary studies receive more citations than primary studies. If this is true, it suggest that citations to secondary studies should be weighted differently than primary studies with respect to scientific merit. Also, we would expect that secondary studies receive more citations than primary studies since they are type of "review" papers and many researchers would find it suitable to cite them to provide an overview of a given testing sub-field.

3.2. Research process

The process that lies at the basis of this study is outlined in Fig. 1, which consists of three phases:

- Article selection (Section 4)
- Development of the systematic map (Section 5)
- Systematic mapping (Section 6)

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

4. Article selection

According to the research process of this study (Fig. 1), the first phase of our study was article selection. For this phase, we followed the following steps in order:

- Source selection and search keywords (Section 4.1)
- Quality assessment (Section 4.2)
- Application of inclusion and exclusion criteria (Section 4.3)
- Finalizing the pool of articles and the online repository (Section 4.4)

4.1. Source selection and search keywords

Based on the SM guidelines [2,16], to find relevant studies, we searched the following two major online search academic article search engines: Scopus and Google Scholar. These two were deemed as enough since Scopus and Google Scholar covers all major publisher venues, e.g., Elsevier, Springer, ACM and IEE publications. Our search keyword in both search engines was:

```
("Systematic mapping" OR Mapping OR
"Systematic literature review" OR "Systematic
review" OR "Literature review" OR Review OR
Survey)
AND (Testing OR Validation OR Verification)
AND Software
```

The searches were conducted using the above keywords in the paper titles and abstracts. Scopus has a strict search engine and returned exactly–matching items. When using the Google Scholar, we utilized its relevance ranking approach (e.g., Google's PageRank algorithm) to restrict the search space. For example, when we applied the above search string to Google Scholar, 2,790,000 results would show as of this writing (December 2015), but as per our observations, relevant results usually only appear in the first few pages. Thus, we checked the first several pages (i.e., we somewhat observed a search "saturation" effect) and only continued further if needed, e.g., we proceeded to the $(n+1)$th page only if the results in the nth page still looked relevant.

The initial search phase yielded 117 papers. To decrease the risk of missing relevant studies, similar to previous SM and SLR studies, we conducted selected snowballing [35] (forward and backward) as well, in which we found 6 additional papers as shown in Fig. 1. All studies found in the additional venues that were not yet in the pool of selected studies but seemed to be candidates for inclusion were added to the initial pool. With the above search strings and search in specific venues, we found 123 studies which we considered as our pool of potentially-relevant studies and were then voted on (also depicted in Fig. 1). At this stage, studies in the pool were ready for application of inclusion/exclusion criteria as described next. We should note that already in this stage we performed initial screening of the paper to reduce the potentially large amount of totally non-relevant entering the pool. For example, we did not include in the pool the obviously non-relevant papers with titles such as: "*Flow volume and shunt quantification in pediatric congenital heart disease by real-time magnetic resonance velocity mapping-a validation study*".

4.2. Quality assessment

Each candidate SM and SLR was evaluated using the same set of criteria adopted by previous research studies (e.g., by Kitchenham) et al. in tertiary studies [6,7,13]. This set of criteria was defined by the Centre for Reviews and Disseminations (CDR) Database of Abstracts of Reviews of Effects (DARE), of the York University [36]. The criteria are based on four quality assessment questions:

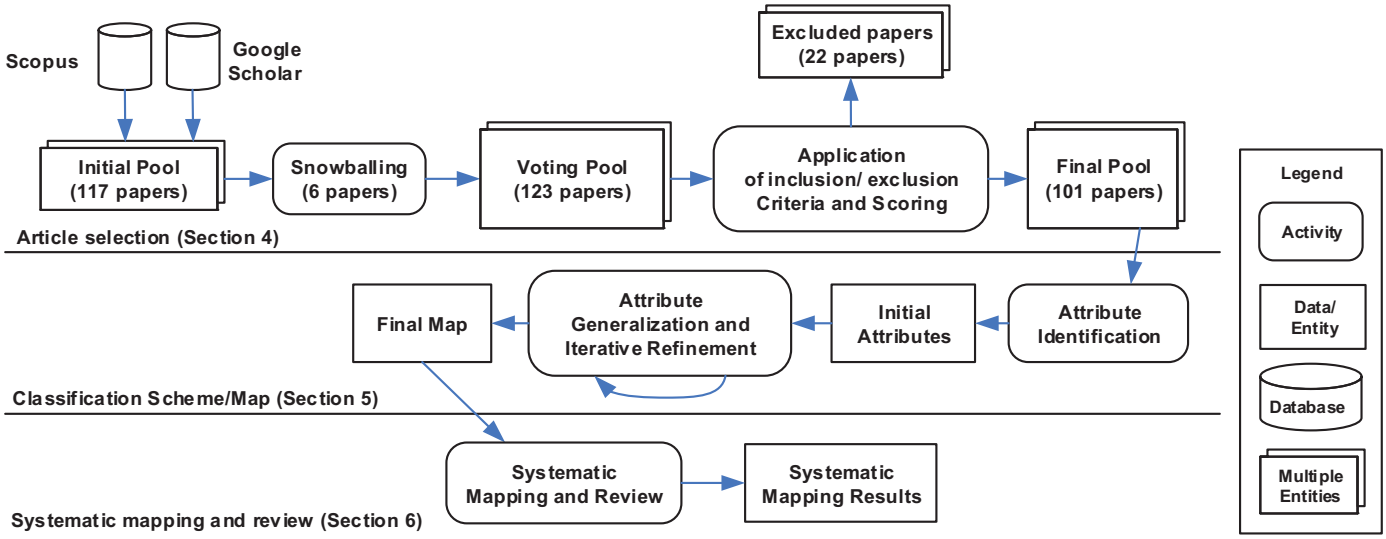


Fig. 1. Research process used to conduct this study.

Table 3
Systematic map developed and used in our study.

RQ	Focus	Types, categories and metrics	Multiple (M) or single (S) value
RQ1: What software testing specific areas that have been investigated in the secondary studies?	Software testing method	{Model-based, mutation, search-based, combinatorial, symbolic execution, random, other}	M
	Type of System Under Test (SUT)	{SOA / web-service, product-line, GUI, web-application, component-based software, aspect-oriented software, cloud, protocol, mobile, concurrent}	S
	Software testing type/phase	{Regression, unit, integration, system, acceptance testing}	M
RQ2: What types of research questions are being investigated?	Types of RQs	As proposed in [34]: {Existence, description and classification, descriptive-comparative, frequency distribution, descriptive-process, relationship, causality, causality-comparative, causality-comparative interaction, design}	M
RQ3: What are the annual demographics of the secondary studies	Number of secondary studies	Integer	S
	Types of secondary studies	{SM, SLR, regular survey, other}	S
	Size of pools of primary studies	Number of primary studies after inclusion/ exclusion as an integer	S
	Quality of secondary study	The four criteria defined and proposed in [36] as discussed in Section 4.2. Each criterion is assigned a value from the set {0, 0.5, 1} and the sum is in range of 0…4.	S
	Number of RQs	Integer	S
RQ4: What are the highest cited secondary studies and are the secondary studies cited more often than primary studies?	Annual publication trend versus primary studies	Year of publication as an integer	S
	Citations to secondary studies	Number of citations as an integer, extracted from Google Scholar on Dec. 1, 2015	S

1. Are the review's inclusion and exclusion criteria described and appropriate?
2. Is the literature search likely to have covered all relevant studies?
3. Did the reviewers assess the quality/validity of the included studies?
4. Were the primary data/studies adequately described?

The above questions were scored as proposed by Kitchenham et al. [6,7]. For each candidate secondary study in our pool, the quality score was calculated, by assigning {0, 0.5, 1} to each of the four questions and then adding them up.

4.3. Application of inclusion/exclusion criteria

In our study, the following inclusion criteria were considered:

1. The topic of each secondary study should be software testing
2. Whether the study was peer reviewed.
3. The quality score of the secondary study (as discussed in Section 4.2) was at least 2 out of 4.

If multiple studies with the same title by the same author(s) were found, the most recent one was included and the rest were excluded. Only studies written in English language and only the ones which were electronically available were included. If a conference study had a more recent journal version, only the latter was included. Recent tertiary studies in SE, e.g., [6,7], only included SLR and SM studies in their analysis, since SM and SLR studies are usually considered more systematic and rigorous compared to regular surveys. However, since we have seen highly-cited and popular survey papers such as [37–41], we decided to include survey papers in our pool as well. In terms of scope, we only included secondary studies on software testing and excluded those on software quality, e.g., [42].

We should also mention that the online repository of the papers in our pool (hosted on Google Docs) [43], refer to the tab "Excluded" in the online spreadsheet, contains detailed explanations on why each paper has been excluded from the primary pool. We discuss several examples of excluded articles and the reasons for their exclusion. Secondary studies cited in [44,45] were excluded since they had low level of comprehensiveness, i.e., they were only 3.5 and 2.5 pages long respectively. A 2002 conference paper cited as [46] was excluded since a newer 2004 journal version [39] of it was added to the pool. Finally, [47] was excluded since its survey component was short.

4.4. Final pool of secondary studies and the online repository

After applying the inclusion/exclusion criteria, the final pool was finalized with 101 studies, and the list has been published using the Google Docs system, along with the systematic mapping of the studies, in an online repository [43]. Also, the bibliographic details for the all the 101 secondary studies are available in another online document [48]. We will refer to the studies in the rest of this paper in the form of [S1], ..., [S101] and these labels are clearly identifiable in both online repositories [43,48].

5. Development of the systematic map

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

5.1. Iterative development of the systematic map

To develop the systematic map, we followed the approach conducted in our recent SM studies [17–20,23–26] and also the guidelines by Petersen et al. [16]. As shown in Fig. 1, 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 relevant to our research project, we recorded them in an online publicly-accessible spreadsheet (hosted in the online Google Docs spreadsheet [43]) to facilitate further analysis. With the relevant studies identified and recorded, 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) type of software testing subjects covered by secondary studies, and (2) types of secondary studies.

We refined these broad interests into a systematic map using an iterative approach. The author conducted an initial pass over the data, and based on (at least) the title, abstract and introduction of the studies, created a set of initial categories and assigned studies to those categories. When the assignment of studies to categories could not be clearly determined just based on the title, abstract and introduction, more of the study was considered. In this process, both the categories and the assignment of studies to the categories were further refined.

5.2. Final systematic map

Table 3 shows the final classification scheme that we developed after applying the process described above. Columns 1 and 2 are self-explanatory. Column 3 denotes the set of all possible values for the attribute, categories and/or metrics used for that RQ. Finally, column 4 indicates for an attribute whether a single selection or multiple selections were made.

5.3. Metrics and data extraction

To extract data, the studies in our pool were reviewed with the focus of each RQ and the required information was extracted according to the systematic map shown in Table 3.

Types, categories and metrics for all RQs except RQ 1 are straightforward and self-explanatory in Table 3. For RQ 1, we extracted as a brief text field (string) the software testing subject of each study, e.g., "test oracles" for [S95] and "model-based testing" for [S12].

In terms of the operational logistics of our data extraction activity, we created an online publicly-accessible spreadsheet on Google Docs [43]. Fig. 2 depicts a screenshot of the spreadsheet in which the data extraction for RQs 1 and 2 are shown. Last but not least, we should mention that we ensured to incorporate as much explicit "traceability" links between our mapping data and the studies in the pool as possible. We have put traceability "comments" inside the cells of the online repository [43] where needed to justify why a given mapping was done (an example is shown in Fig. 2).

6. Results

Results of the study are presented in Sections 6.1–6.4.

6.1. RQ1: software testing areas investigated in the secondary studies

To summarize the software testing subjects covered by the studies, we derived the keywords of each study from its title. To get a high-level picture of the covered subjects, we generated a word cloud of paper titles using an online tool (www.wordle.net), which is shown in Fig. 3. Common/obvious words in this context such as "software", "testing", "systematic mapping", "systematic literature review" and "survey" have been removed from the input text set. The (font) size of each keyword denotes the number of times it has appeared in the titles of the secondary studies.

We can notice that some subjects have been studied more compared to others. For example, search-based and web testing have each been studied the most, i.e., in five secondary studies each, refer to the works in [40,49–52] and [18,19,53–55], respectively.

Furthermore, we performed more detailed qualitative classification of the papers from three dimensions the testing method, the SUT, and the testing phase. We started with these three classes, but otherwise we let the classification rise from the data as is typical qualitative data-analysis, see for example [56] and [57], albeit our existing knowledge of testing research has undoubtedly affected the results. Due to this process, the results paint a somewhat different picture what is shown in Fig. 3. For example, we did not consider test-case generation as a testing method by itself and, thus, papers on test case generation would be classified on what they use as their method for test-case generation which would things like models or search-based techniques.

6.1.1. Testing methods

The testing method states what is used to perform the testing. We only included techniques, such as search-based testing, here and classified more general or vague approaches such as requirements-based testing [S27] to Section 6.1.4. We could also characterize the method papers as solution-oriented papers as they focus on a particular type of solution, e.g. model-based testing, as a solution. Many papers (39) specified something that we could classify as a testing method and Table 4 list all methods with more than two literature reviews.

Model-based testing was the most popular testing method with 15 papers. Some of those papers were focused on a particular area, e.g. UML-model based coverage criteria [S11] while others were

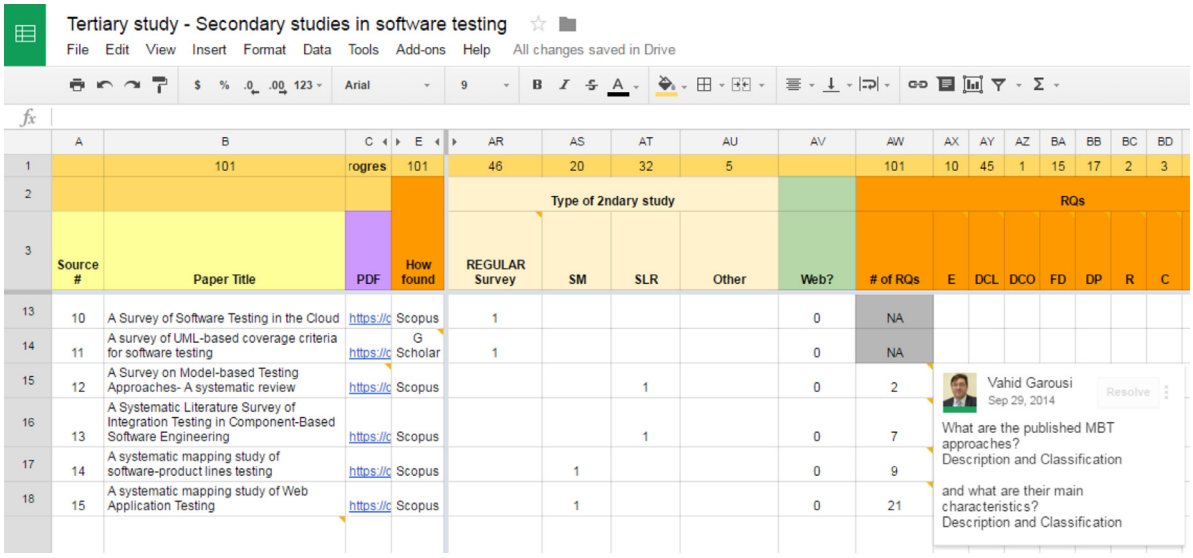


Fig. 2. Operational logistics of our data extraction activity as managed by an online publicly-accessible spreadsheet in Google Docs (Drive) system [43].

Fig. 3. Word cloud of paper titles (using the online tool at www.wordle.net).

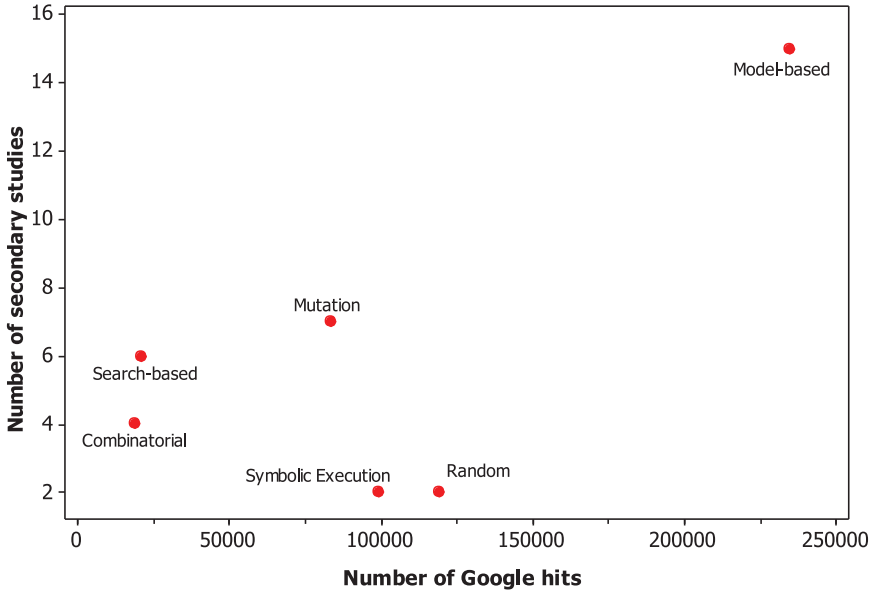


Fig. 4. Scatter-plot of the data in Table 4.

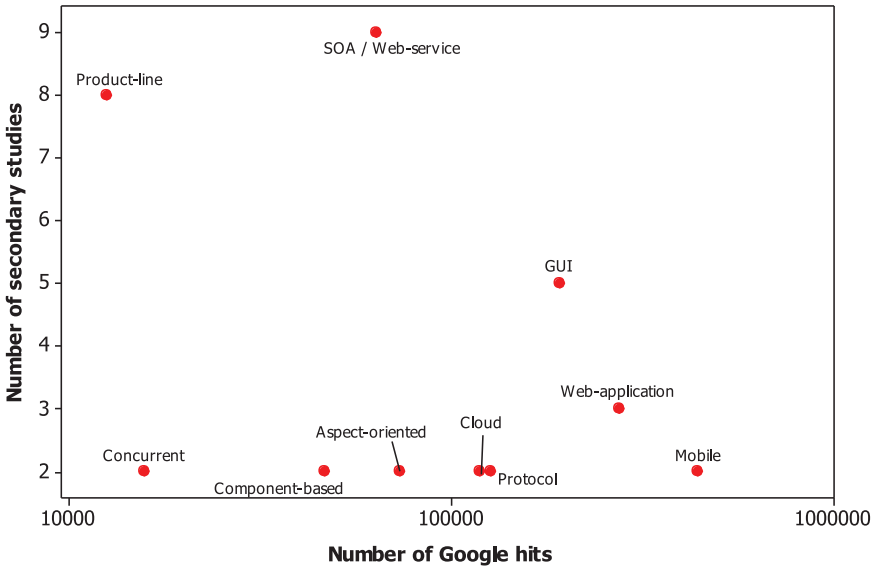


Fig. 5. Scatter-plot of the data in Table 5.

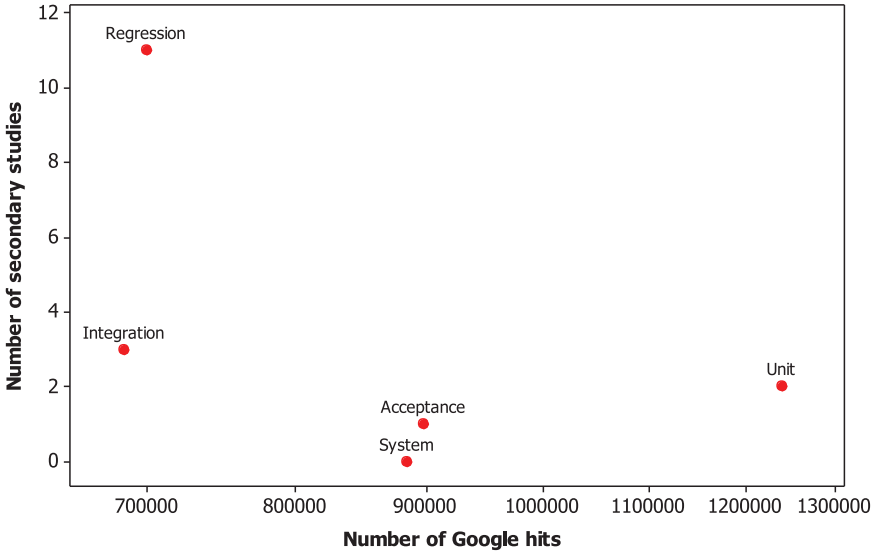


Fig. 6. Scatter-plot of the data in Table 6.

Table 4
Comparison of popularity of software testing methods in Google hits versus # of secondary studies.

Software testing methods	Number of secondary studies	Number of Google hits[a]	References
Model-based	15	235,000	[S11, S12, S19, S22, S24, S30, S35, S41, S44, S55, S61, S68, S78, S82, S97]
Mutation	7	83,600	[S16, S21, S28, S48, S54, S63, S84]
Search-based	67	21,000	[S18, S20, S30, S46, S64, S65, S77]
Combinatorial	4	18,600	[S4, S29, S30, S36]
Symbolic Execution	2	99,400	[S9, S30]
Random	2	119,000	[S30, S77]

[a] Search string was ["<software testing method> + testing" + software]. For example: "mutation testing" software. And for Symbolic Execution we had: "Symbolic Execution" testing software

more general [S12]. Search-based testing and Mutation testing were the next popular methods each with seven papers. Finally, combinatorial testing had four literature reviews, while Symbolic Execution and Random Testing had two literature reviews each.

It is notable that there were no studies focusing purely on manual testing methods performed by humans through a system user interface. Manual testing is still extremely common in industry, according to a recent survey with 1543 responses from the executives of software industry only 28% of test cases are automated [58].

In order to have a reference point of the popularity of testing methods in the general public and in the community, we checked

the number of Google hits[1] for each method as shown in Table 4. Our rationale for comparing the number of Google hits for each topic with the number of studies was to compare the level of attention for each topic in the research community versus its popularity in the general public. We interpret such trends to observe the advancement and popularity of each topic and suggest recommendations.

If the testing method and our approach to derive the search string produced a string that was unlikely to get many hits, we would modify the string to our best ability to find a string that would accurately represent the testing method but produce the largest amount of hits. For example adding the word "software" to our search string was mandatory as searches with string "mutation testing" brought up many irrelevant hits from other fields, e.g., genetics. Still, it is possible that on some instances we might not found the most representative search results.

We found that model-based testing, which was the testing method with the most secondary studies, produced the most Google hits (235,000). Mutation testing had the second most secondary studies, but was ranked only 4th in terms of Google hits. Search-based testing had the 3rd place in number of secondary studies but was ranked only as 5th in the Google hits with 21,000. Random testing had fair amount of Google hits (119,000) despite having only two secondary studies.

Overall, a little relationship exists between the number of Google hits and the number of secondary studies. A partial explanation may lie in the different terms used in the academia and in the industry. Model-based testing is an accepted name on both sides so it gets many secondary studies and many Google hits. However, many of the other techniques are hidden inside tools and industry might not even realize that they are using these techniques. For example, Evosuite[2] has been developed in academic research [59] and it utilizes search-based and mutation techniques, but does not advertise those techniques as it is easier to market the tool with the ability to automatically generate unit-test rather than revealing the technical details of how the tool performs this task.

6.1.2. Systems Under Test (SUT)

The type of System Under Test (SUT) was specified 43 papers. These papers can be characterized as a problem oriented as they are focused on testing problems created by a particular system under test, e.g. product-line testing, that may be solved with several methods. Here we interpret SUT loosely to characterize the type of system or interface that is the target of the testing.

We found SOA[3] / Web-service was the most popular SUT with nine papers. Many of those papers, e.g., [S86, S89, S92], focused on Web-services which is one way to implement SOA. Testing software product lines had eight literature review while five reviews focused on Graphical User Interface testing. Web-application, a special kind of graphical user-interface, was the focus of three papers. Testing Component-based software, Aspect oriented code, Mobile applications, Protocol and Concurrent software all had two papers. Cloud testing also had two papers, it should be noted that both of those papers considered Cloud both as a testing method, i.e. how to use Cloud resources for testing, as well as SUT, i.e. how to test Cloud application, and therefore these papers [S10, S42] are listed in both Tables 4 and 5.

With respect to Google hits we followed a similar approach as with testing methods. We found that the most popular academic topic SOA / Web-service was ranked only as the 7th by Google hits. Furthermore, the SUT that had the second highest number of secondary studies, Product-line, was ranked as the last in terms of Google hits. We are somewhat surprised that GUI testing did not get more Google hits. Although, it is ranked 3rd in popularity with 192,000 hits we expected it to have even more hits. We think the term GUI might be too general as different types of GUIs require different test tools and, thus, the topic is not so popular as we would expect. This is partially confirmed by the popularity of Web-application testing, which includes a particular GUI, that was the second most popular SUT type with 274,000 hits. Finally, as expected Mobile testing has 442,000 hits and that makes it the most popular SUT, and there were found two secondary studies on the topic.

6.1.3. Testing types/phases

Third, 15 papers focused on a particular phase of testing, e.g. unit-testing. From these papers, 11 focused on Regression testing, 3 on Integration testing and 2 on Unit testing see Table 6. It should be noted that none of TDD secondary studies [S25, S39, S43], were purely focused on Unit-testing phase as they had covered papers where TDD had been practiced on System testing phase as well. Yet, for the majority TDD is still considered as unit-testing phase practice even though the idea can be applied in other test phases as well.

For Table 6, we also added testing phases with 1 or no secondary studies as there is in principle a limited set of testing phases, which is not the case for the testing methods or targets. For testing methods and SUTs the search space is much larger. Based on Google hits Unit-testing is the most popular ones with 1,240,000 hits. From Table 6, we can see that System testing has the second highest number of hits with 898,000, but no secondary studies focusing solely on that topic have been performed. We have to note that often the System testing phase is implicit within many papers. Similarly, many papers did not state the testing phase their method would have been the most suitable.

6.1.4. Other factors

Finally, we could not classify all papers to any of the previous categories. As all paper must have a focus within the large software testing domain, we classified and coded them in to the category "Other" factors. The following focus areas were covered in those papers. First, a particular process under which testing is performed was the focus area for six papers. Three of the papers focused on Test-Driven Development [S25, S39,S43] that is a popular approach were tests are developed before the actual code. Two paper focused on testing under particular software development processes: Agile software development [S26], Rapid Releases [S49]. Finally, one paper had reviewed literature of the software testing processes [S85].

Second, the empirical evidence or testing experiments were the focus of four papers: [S16, S20, S63, S99]. Third, test-code engineering or maintaining tests were the focus of two papers: [S67, S71]. Fourth, the use of Cloud computing resources was the topic of two papers [S10, S42]. Finally, single papers were found on the Oracle problem which related to all testing methods [S95], decision on whether to use automated or manual testing [S34], the alignment of requirements and testing [S27], combining static and dynamic QA [S17], knowledge management of software testing [S52], the adherence to standards in testing [S72], and effort reduction of QA [S57].

6.2. RQ2: research questions being investigated

We classified all the RQs using the RQ classification presented by Easterbrook et al. [34] as shown in Table 7. This classification

[1] We realize that this method is far from perfect, however, it is nearly free in terms of effort spent. Thus, we believe that it provides a good analysis in terms of cost and benefits.

[2] http://www.evosuite.org

[3] Service-oriented architectures

Table 5
Comparison of popularity of software SUTs in Google hits versus # of secondary studies.

Type of SUT	Number of secondary studies	Number of Google hits[a]	References
SOA / Web-service	9	63,700	[S22, S60, S66, S86, S87, S89, S92, S98, S101]
Product-line	8	12,600	[S2, S7, S14, S50, S69, S70, S74, S79]
GUI	5	192,000	[S24, S25, S45, S75, S81]
Web-application	3	274,000	[S15, S96, S100]
Component-based software	2	46,700	[S13, S91]
Aspect-oriented software	2	73,000	[S21, S80]
Cloud	2	119,000	[S10, S42]
Protocol	2	127,000	[S5, S56]
Mobile	2	442,000	[S32, S94]
Concurrent	2	15,800	[S38, S62]

[a] Search string was "<SUT> testing" software, e.g., "GUI testing" software. Except for SOA / Web-services, we only used SOA as it gave the largest number of hits For Component-based software, we used "component-based software" testing. For Aspect-based software, we used "Aspect-based software" testing. For Mobile, we used "mobile application testing" software.

Table 6
Comparison of popularity of software testing phases in Google hits versus # of secondary studies.

Software testing phase	Number of secondary studies	Number of Google hits[a]	References
Regression	11	700,000	[S16, S23, S51, S53, S58, S59, S60, S61, S63 S76, S83]
Integration	3	686,000	[S13, S68, S91]
Unit	2	1,240,000	[S1, S98]
Acceptance	1	898,000	[S31]
System	0	885,000	

[a] Search string was "<software testing phase> testing" software, e.g., "regression testing" software.

has been used in other tertiary studies as well, e.g., [8]. Each RQ was classified under one category only. For brevity and space constraints, we are not reporting the entire list of RQs that we have extracted from all studies, but they can be found in the online Google Docs repository [43]. We believe that reviewing the list of RQs in Table 7 and also the full list in [43] can help researchers in raising RQs for new secondary studies and also to raise better, more systematic RQs.

We can see that descriptive-comparative RQs are the most popular by large margin. RQs of this type are mostly investigated in mapping studies which are in SMs and most of the SLRs. The second and third most frequent RQs are frequency distribution (FD) and descriptive-process (DP).

We found that the RQs ranged from well-defined causality question aimed at investigating the effectiveness of certain treatment, as in the "traditional form" of SLRs, to broad exploratory RQs aimed at mapping the current status of research and practice in a given topic. We see in both that there is a shortage or lack of RQs in types towards the bottom of the classification scheme. For example, among all the studies, no single RQ of type Causality-Comparative Interaction or Design was raised. There is a need for such RQs in future studies in this domain.

To get a rough measure of the scale of empirical and evidence-based SE in the studies, we searched for the terms "empirical" and "evidence" in the list of all RQs. 15 and 15 RQs (or about 5.6% of all 267 RQs), respectively, included the two terms. In terms of distinction between SM and SLRs, according to Kitchenham et al. [6,7], any SLR that possesses exploratory RQs and aims to obtain a general view about a research topic, is considered a SM instead. However, Kitchenham also states that this distinction between SM and conventional SLRs can be somewhat fuzzy. Despite having exploratory questions like: "What do we know about the topic T?", which are different from SLRs conventional research questions, we saw that some SM studies provide a more detailed description about the analyzed studies.

Finally, we observed that some studies have raised multiple RQs into one, e.g. [S12] posed the following two questions as one single RQ: "*What are the published MBT approaches and what are their main characteristics?*". [S18] raised one single question as follows which is in fact two RQS combined into one: "*In which non-functional testing areas have search-based techniques been applied and what are the different meta-heuristics used?*"

As shown in Table 7, to our surprise, there was no RQs in any of the secondary studies of type Causality-Comparative Interaction (CCI) nor Design (D). One could identify these RQs as more sophisticated ones compared to the others, e.g., a CCI RQ may look like this: "*Does test approach A or B cause more defect detection under one condition but not others?*". We hope to see secondary studies with such RQs in future.

6.3. RQ3: annual trends of types, quality, and number of primary studies

6.3.1. Annual trends and types

The annual trend of the secondary studies included in our pool is shown in Fig. 7(a). The first survey was published in 1994 [S56]. There have been no secondary studies from 1997 until 2002. The major wave of the studies starts from 2007 and grows quickly until year 2012 after which there is a decline. Note that the count for the year 2015 is partial since this study was conducted in the Fall of 2015 and more studies are expected to appear in the rest of 2015. As per the trend in Fig. 7, we expect to see relatively high numbers of studies in this area in near future.

We also wanted to compare the annual trend of SLRs in our pool versus all SLRs in SE. Fig. 7(b) shows that comparison (note that data are cumulative). The latter data has been taken from a 2013 tertiary studies in SE [14], but only includes the data until 2011. As we can see, the SLRs in testing favorably constitute a good portion of the SE SLRs. To the best of our knowledge, the first SLR in testing appeared in 2007 and was on model-based testing [61].

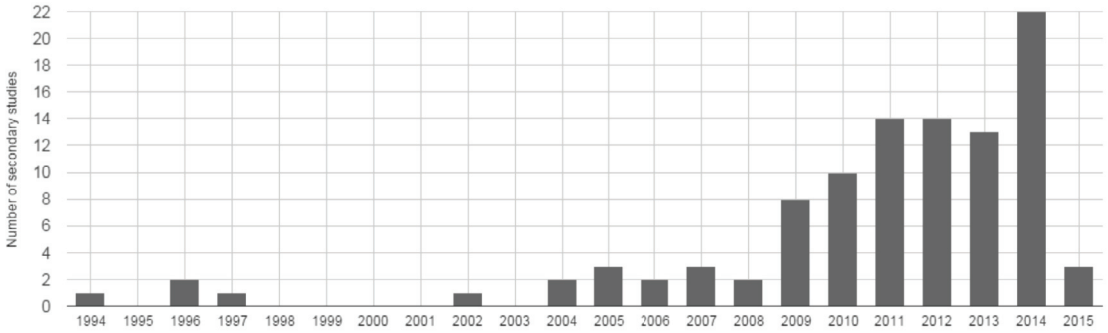
In terms of secondary study types, there are 20 SMs, 32 SLRs and 46 regular surveys in the area. There were also five studies under the 'other' type, as phrased by the researchers: who had conducted the studies: [S30] is an "orchestrated" survey, [S49] is a semi-SLR, [S53] is a critical review, [S58] is a critical evaluation, and [S99] is a meta-analysis. An orchestrated survey is a

Table 7
A classification scheme for RQs as proposed by [34], number of studies under each type, and examples of RQs from the SMs and SLRs for each.

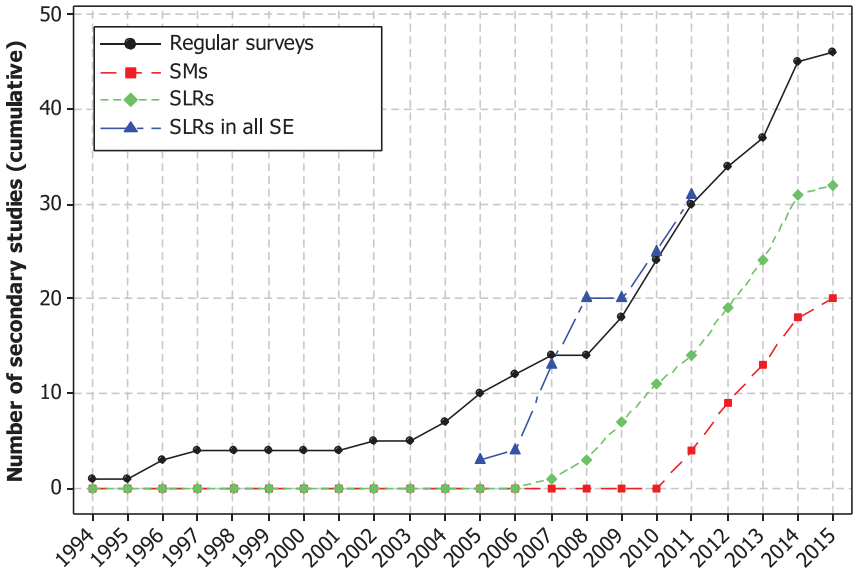
RQ category	Sub-category	RQ code	# of studies	# of RQs in the pool	Example RQs
Exploratory	Existence	E	10	16	Does X exist? • Do the approaches in the area of product lines testing define any measures to evaluate the testing activities? [S2] • Is there any evidence regarding the scalability of the meta-heuristic in the area of search-based test-case generation? [S20] • Can we identify and list currently available testing tools that can provide automation support during the unit-testing phase? [S1]
	Description and classification	DCL	45	171	What is X like? • Which testing levels are supported by existing software-product-lines testing tools? [S2] • What are the published model-based testing approaches? [S12] • What are existing approaches that combine static and dynamic quality assurance techniques and how can they be classified? [S17]
	Descriptive-Comparative	DCO	1	1	How does X differ from Y? • Are there significant differences between regression test selection techniques that can be established using empirical evidence? [S23]
Base-rate	Frequency distribution	FD	15	41	How often does X occur? • How many manual versus automated testing approaches have been proposed? [S15] • In which sources and in which years were approaches regarding the combination of static and dynamic quality assurance techniques published? [S17] • What are the most referenced studies (in the area of formal testing approaches for web services)? [S22]
	Descriptive-process	DP	17	29	How does X normally work? • How are software-product-lines testing tools evolving? [S2] • How do the software-product lines testing approaches deal with tests of non-functional requirements? [S1] • When are the tests of service-oriented architectures performed? [S87]
Relationship	Relationship	R	2	2	Are X and Y related? • Is it possible to prove the independence of various regression-test-prioritization techniques from their implementation languages? [60] (let us recall the template for this RQ type: "Are X and Y related?")
Causality	Causality	C	3	3	Does X cause (or prevent) Y? • How well is the random variation inherent in search-based software testing, accounted for in the design of empirical studies? [S20] • How effective are static analysis tools in detecting Java multi-threaded bugs and bug patterns? [S38] • What evidence is there to confirm that the objectives and activities of the software testing process defined in DO-178B provide high quality standards in critical embedded systems? [S72]
	Causality-comparative	CC	4	4	Does X cause more Y than does Z? • Can a given regression-test selection technique be shown to be superior to another technique, based on empirical evidence? [S23] • Are commercial static-analysis tools better than open-source static-analysis tools in detecting Java multi-threaded defects? [S38] • Have different web-application-testing techniques been empirically compared with each other? [S100]
	Causality-comparative interaction	CCI	0	0	Does X or Z cause more Y under one condition but not others? • There are no such RQs in the pool.
Design	Design	D	0	0	What's an effective way to achieve X? • There are no such RQs in the pool.
		Sum:		267	

collaborative work usually among a team of researchers collecting self-standing sections, each focusing on a key surveyed topic, in the case of [S30] a test generation technique. Each section is independently authored by a world-renowned active researcher (or researchers) on the topic. Fig. 7 shows the cumulative trend of types of secondary studies.

Albeit the systematic nature of SM and SLR studies and the fact that they are generally considered higher-quality sources to study and refer to (compared to regular surveys), regular surveys are still being conducted and published. The first two regular surveys in this area [37,62] were published in as early as 1996. However, the first SM [49] and SLR [61] in the area were not published until 2008 and 2007, respectively.



(a)



(b)

Fig. 7. Number of and types of secondary studies.

6.3.2. Quality, number of primary studies and number of RQs

The number of primary studies studied in each secondary study varies. A large number of regular surveys did not explicitly report their number of primary studies and, in those cases, we counted the number of references at the end of those papers, as an estimate of the size of paper pools. We visualize the data in Fig. 8(a) which visualizes the number of primary studies analyzed in each secondary study versus its publication year. A regression fit curve is also provided. The Pearson correlation coefficient of the data in the two axes is 0.13, denoting that the size of pool of primary studies is slowly increasing as the years go by. This is perhaps mainly due to the fact that as years go by, more and more primary studies in various areas of software testing get published, and thus, secondary studies in this area would include a large pool of primary studies to analyze and review. The three studies with the largest pool sizes in each of the three types of secondary studies are:

- [S89] for SM with 150 papers with focus on testing web services
- [S12] for SLR with 202 papers with focus on model-based testing
- [S95] for Regular surveys with 611 primary with focus on test oracles

As discussed 4.2 (quality assessment), each SM and SLR was evaluated using a set of quality-related criteria used in earlier tertiary studies [6,7,13]. This set of criteria was defined by the Centre for Reviews and Disseminations (CDR) Database of Abstracts of Reviews of Effects (DARE) of the York University [36]. The criteria are based on four quality assessment questions:

1. Are the review's inclusion and exclusion criteria described and appropriate?
2. Is the literature search likely to have covered all relevant studies?
3. Did the reviewers assess the quality/validity of the included studies?
4. Were the primary data/studies adequately described?

The above questions were scored as suggested by Kitchenham et al. [6,7]. For each study in our pool, the quality score was calculated by assigning {0, 0.5, 1} to each of the four questions and then adding them up, thus having a value in the range of (0, 4). Fig. 8(b) shows, as an individual-value plot, the annual trend of total quality score of SMs and SLRs, including average bars. As we can observe, the average quality score have been between 2 and 3 (out of 4) until 2014, but it has recently increased slightly Fig. 8(b).

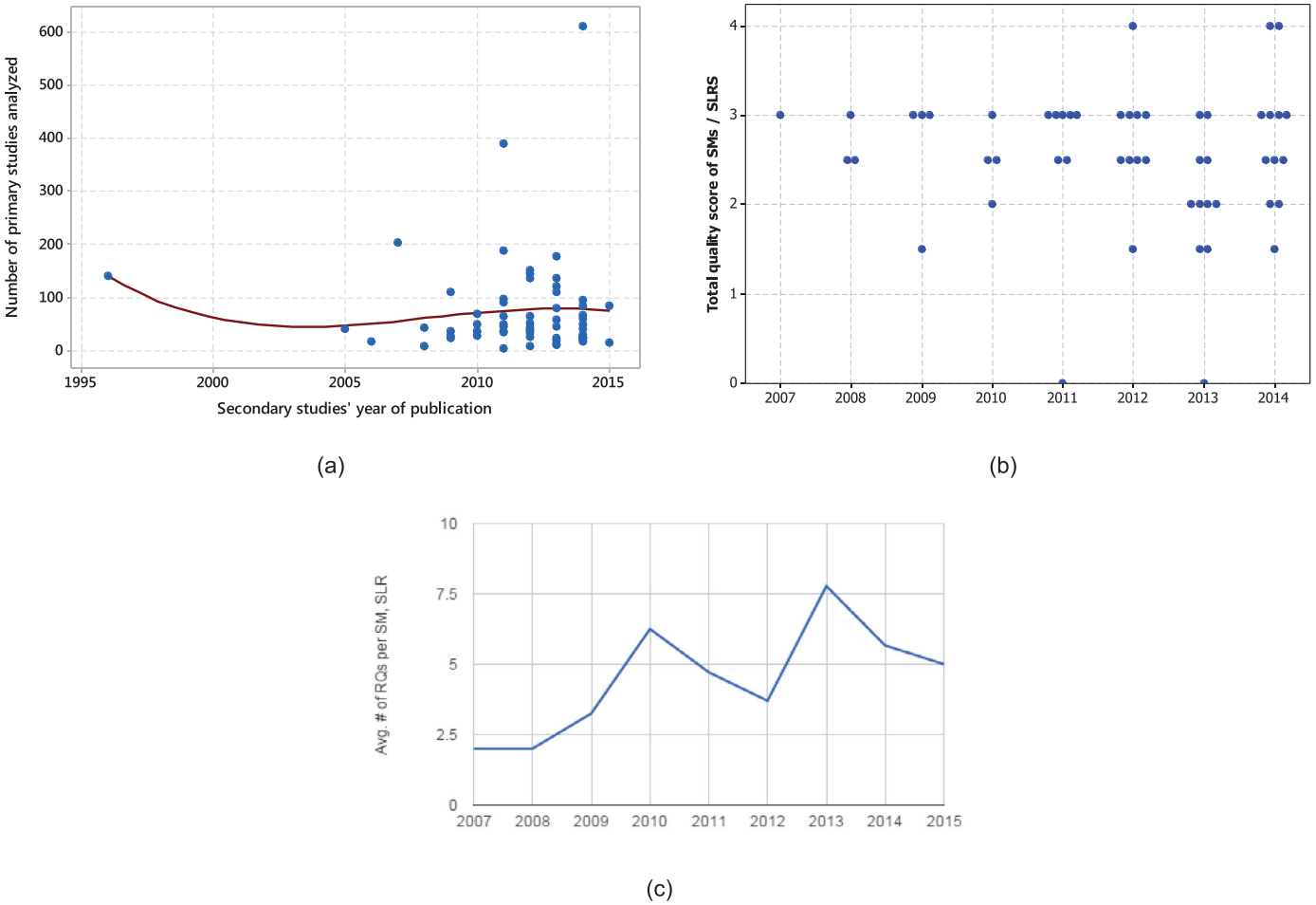


Fig. 8. Quality, number of primary studies and number of RQs across the years.

According to Kitchenham et al. [6,7], SM and SLR studies should raise proper and useful RQs. In our analysis of the RQs, only SM and SLR studies were included. Regular surveys were excluded. We counted the number of RQs in each study and considered it as a rough measure of comprehensiveness of each study. However, we do realize that the number of RQs are largely affected by how ones structures the presentation of the results and whether one uses main RQ sub RQ type of structure. In total, all the 14 SMs and 21 SLRs together investigated 190 RQs. On average, a SM or a SLR study had 5.4 RQs. The two studies with the highest number of RQs were: a SM of web-application testing [18] (with 21 RQs) and a SM of graphical-user-interface testing [17] (with 20 RQs). Surprisingly we found two SLR studies which had not formally posed any RQs, but had rather conducted the review using an exploratory approach. They were: (1) a SLR on fault-based mutation testing techniques and tools for Aspect-J [63], and (2) a SLR on automated acceptance testing [64]. As we can notice Fig. 8(c), the average annual numbers of RQs are gradually increasing, e.g., it reached 9 RQs in 2013. This suggest that SM and SLRs in this area are becoming more comprehensive.

6.4. RQ4: highest-cited secondary studies, and citation comparison of secondary versus primary studies

6.4.1. Highly-cited secondary studies

Studying highest-cited papers in software engineering is becoming a popular topic, e.g., [65-67]. We were also curious to find the highest-cited secondary studies. To investigate number of citations we used two types of citation metrics for this purpose: (1) Absolute (total) number of citations to a paper since its publication, and (2) Average number of citations per year since publication of a paper. Compared to the total number of citations, the normalized metric essentially returns the average number of citations of a paper per year, since its publication year. Total and average-annual numbers of citations versus publication year of the secondary studies are shown as two scatter plots in Fig. 9, in which each chart includes 101 points corresponding to 101 secondary studies. The Y axes are shown in the log-scale for better visualization.

The lists of top-5 highly-cited secondary studies based on each of the two metrics are shown in Tables 8 and 9. [S55] and [S64] are, respectively, the two highly-cited secondary studies based on the two metrics. [S55] is a survey on testing finite state machines published in 1996, and [S64] is a survey on search-based software test data generation. As we can see, in both top-5 lists, 9 out of the 10 cases are surveys while only one is an “orchestrated” survey. Thus, it seems that regular surveys are cited more in general compared to SLR and SM studies.

We should note, in the outset of our study, that reputation of the authors of a given paper could be a factor our analysis above. But quantifying reputation is not easy and discussing factors impacting the citation of a paper is outside the scope of our current paper and has been discussed in other studies, e.g., [68–70]. We thus only had two sources of quantified data to base our analysis upon: absolute and normalized # of citations, as we did in our recent bibliometric studies, e.g., [65,71–73].

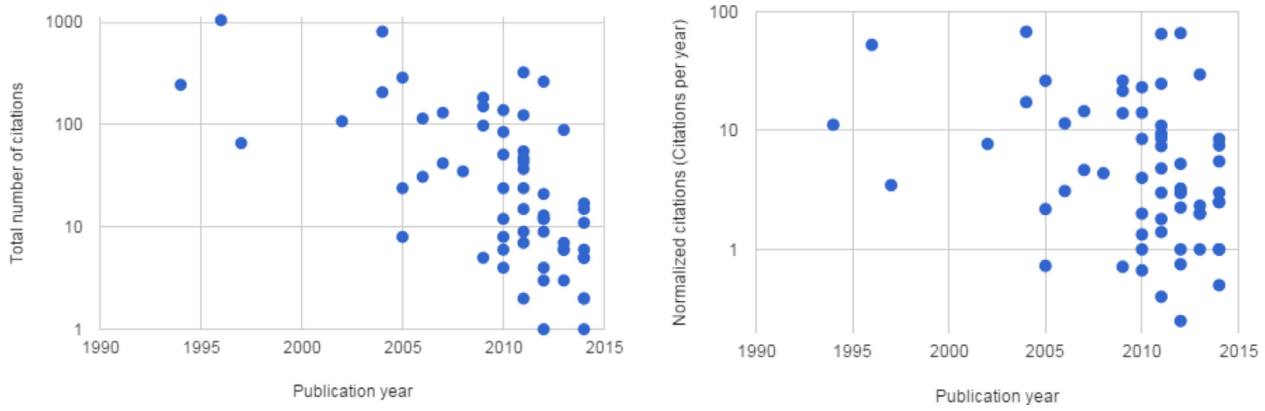


Fig. 9. Total and normalized average annual number of citations and of citations versus publication year of the secondary studies.

Table 8
Top-5 highly-cited secondary studies based on total number of citations.

Reference	Paper title	Secondary study type	Publication year	Total number of citations	Average-annual number of citations
[S55]	Principles and methods of testing finite state machines: a survey	Survey	1996	1052	52.6
[S64]	Search-based software test data generation: a survey	Survey	2004	814	67.8
[S28]	An analysis and survey of the development of mutation testing	Survey	2011	325	65.0
[S36]	Combination testing strategies: a survey	Survey	2005	289	26.2
[S59]	Regression testing minimization, selection and prioritization-a survey	Survey	2012	264	66

Table 9
Top-5 highly-cited secondary studies based on average-annual number of citations.

Reference	Paper title	Secondary study type	Publication year	Total number of citations	Average-annual number of citations
[S64]	Search-based software test data generation-a survey	Survey	2004	814	67.83
[S59]	Regression testing minimization, selection and prioritization-a survey	Survey	2012	264	66
[S28]	An analysis and survey of the development of mutation testing	Survey	2011	325	65
[S55]	Principles and methods of testing finite state machines-a survey	Survey	1996	1052	52.6
[S30]	An orchestrated survey of methodologies for automated software test case generation	Orchestrated survey	2013	89	29.67

Table 10
Descriptive statistics of the citation data for regular surveys, SMs and SLRs.

Metrics	Statistics	Citation data sets		
		Regular surveys	SMs	SLRs
All citations	**Average**	100.8	13.6	24.0
	Median	27.5	3.0	6.0
Average citations per year	**Average**	11.1	3.0	4.4
	Median	4.11	1	1.3

Table 11
p-values of the Mann–Whitney test on the four normalized citation data sets for regular surveys, SMs and SLRs.

	SMs	SLRs
Regular surveys	0.009	0.014
SMs	–	0.75
SLRs	–	–

6.4.2. Comparing citations to regular surveys, SMs and SLRs

To our surprise, we found that regular surveys are cited far more often than SMs or SLRs. The descriptive statistics in Table 10 show that the average number of citations for regular surveys is 11 per year while for SMs and SLRs numbers are much lower, only 3.0 and 4.4, respectively. Other descriptive numbers in the table as well as Fig. 10 confirm this relationship. The statistical tests on Table 11 show that this difference is statistically significant. We discuss the possible reasons for this in the Discussions section (Section 7)

6.4.3. Comparing citations between secondary and primary studies

We wanted to compare citations to secondary studies versus citations to primary studies. One would expect that since secondary studies provide overview and trends on a subject area, more researchers read and cite to them, more compared to primary studies. Other authors have also worked on similar topics, e.g., [74].

For baseline analysis, we considered three other citation data sets that we had access to (i.e., the first author was involved in): (1) citations to the pool of papers in web testing [18], (3) GUI testing [17] and (4) UML-driven software performance engineering (UML-SPE) [25]. The latter citation data set (UML-SPE) has been

Table 12
Descriptive statistics of the four citation data sets.

Metrics	Statistics	Citation data sets			
		Secondary studies	Web testing	GUI testing	UML-SPE
All citations	**Average**	57.2	31.9	21.1	34.9
	Median	9	12	7	18.5
Average citations per year	**Average**	7.3	5.5	3.5	4.7
	Median	2.5	3	2	2.6

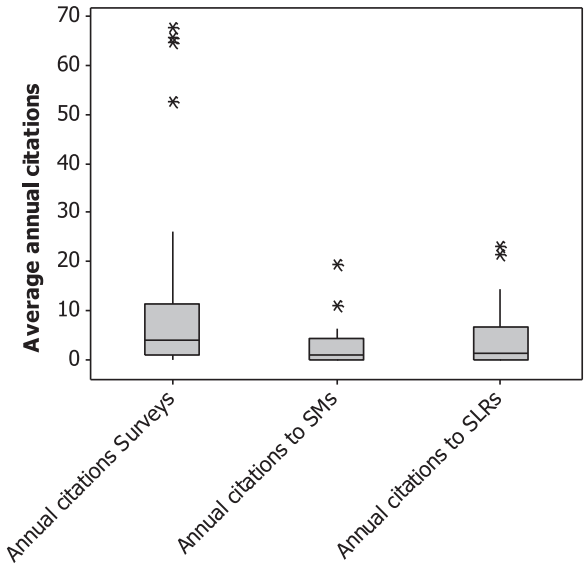


Fig. 10. Box-plots of average normalized citations to regular surveys, SMs and SLRs.

included as a data set in a non-testing subject area to enable out-of-topic comparison of citation trends.

We calculated and show in Table 12 the statistics of the four citation data sets. We can see that, in terms of both citation metrics (total number and average annual number of citations), citations to the secondary studies are higher than the papers in the pool of all three SM studies (web testing [18], GUI testing [17] and UML-SPE [25]). This suggests that the research community has already recognized the value of secondary studies. However, based on the Mann–Whitney statistical test results (shown in Table 13), the difference for none of the data sets is statistically significant.

We visualize in Fig. 11, as individual-value plots, values of both metrics for four citation data sets: (1) all 58 secondary studies analyzed in this article, (2) web testing [18], (3) GUI testing [17] and (4) UML-SPE [25]. All of the four distributions in both the individual-value plots are skewed toward the bottom, denoting that both total and also yearly-average citations are generally low for most of the papers in each of the four pools of papers.

7. Discussions

Section 7.1 discusses the implications of this tertiary study. Section 7.2 discussed potential threats to the validity of our study and steps we have taken to minimize or mitigate them.

7.1. Interpretations and implications

7.1.1. Implications for the industry (ISTQB syllabus)

Figuring out the current level of testing knowledge in industry is not trivial as there are large fluctuations of knowledge level among test practitioners and organizations. We used the ISTQB foundation level syllabus [75] as a representative baseline of

Table 13
p-values of the Mann–Whitney test on the four normalized citation data sets.

	Web testing	GUI testing	UML-SPE
Secondary studies	0.38	0.28	0.41
Web testing	–	0.20	0.83
GUI testing	–	–	0.14
UML-SPE	–	–	–

industry's knowledge and expectations in testing and mapped our results back to this syllabus, as shown in Table 14. This syllabus undoubtedly has large impact on educating testers in the industry as according to ISTQB, over 400,000 certificates based on the syllabus have been awarded as of 2016 [76]. We recognize that ISTQB has several additional syllabuses called the foundation extensions, advanced, and expert levels [77]. Detailed comparison to those is beyond the scope of this work. We have chosen the foundation level as it is first the one taken by each tester, thus, it is also the most widely used. We present two types of implications: (1) Areas that should be included or improved in the ISTQB syllabus; and (2) Areas where new secondary studies are needed to provide empirical evidence to the ISTQB syllabus.

Areas that should be included in ISTQB are testing techniques which have gained high academic interests namely: combinatorial testing (pair-wise testing), search-based testing which currently is mostly popular in white-box testing, and mutation testing (see Section 6.1.1 for references). The test oracle problem should also be covered in the ISTQB syllabus in more detail, see [S95]. Testing using cloud computing resources could also be considered as part of the ISTQB syllabus [S10, S42]. Additionally, testing based on software requirements [S27] and the use of standards [S72] in testing could be more coherent as currently references to standards and requirements are scattered several places in the ISTQB syllabus.

Several areas need to be covered by future secondary studies to support ISTQB syllabus, as listed next:

- Test management, organization and monitoring did not have any secondary studies, perhaps due to not having enough primary studies. This topic lies in the intersection of two research disciplines software engineering and management sciences. Valuable secondary studies could be made for example on the role of independence in software testing, test exit criteria used in the industry, and the role of product risk in test management.
- Two of the ISTQB's seven principles of testing need more academic studies, e.g.., Principle 3: early testing, and Principle 5: "pesticide paradox".[4] Furthermore, we think ISTQB should modify Principle #4 (Defect Clustering) to include the lat-

[4] "If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. To overcome this Pesticide Paradox, it is really very important to review the test cases regularly and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects" [68].

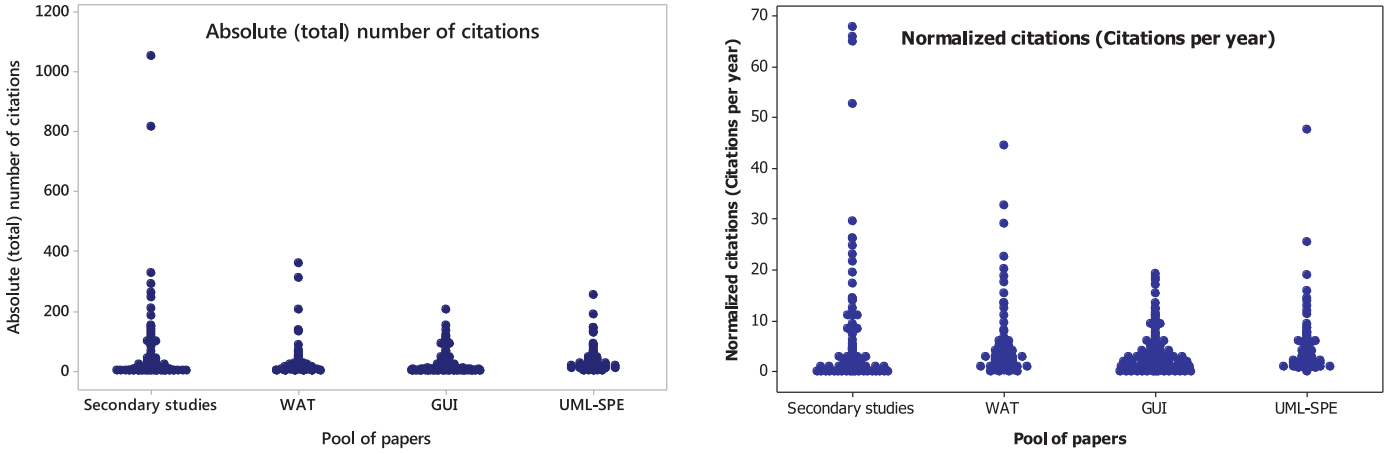


Fig. 11. Individual-value plots showing the number of citations to the four pools of papers.

est empirical knowledge of defect prediction, e.g., [78], as it currently contains information with questionable value.
- Human factors or psychology of testing as it is called in ISTQB syllabus also raise for secondary studies in this area. For example, a tester's mindset, or the best ways to provide criticism in an organization are topics testers' faces on a daily basis in their work and there are not many academic studies on these topics.
- Experience-based testing techniques or exploratory testing is also an area that exists in the syllabus but has no secondary studies.
- Several papers on test automation and automation tools were found, but no paper focused in detail on the test automation tool adaption. This area becomes increasing important as the need for software test automation increases due to pressure to perform rapid or continuous releases.

7.1.2. Implications for the software engineering education (based on SWEBOK guide)

To provide implications for software engineering education, we mapped our findings to the Software Testing chapter of the Guide to the Software Engineering Body of Knowledge (SWEBOK) [81]. For example, the ACM graduate curriculum guidelines [82], is based on SWEBOK v3.0. It is difficult to quantify the impact of SWEBOK, however, it should form a common basis for the university education of software engineers worldwide. Thus, supporting SWEBOK with scientific studies can have far reaching implications. Similar to the previous section, we present two types of implications for this case as well: (1) Areas that should be included or improved in the SWEBOK; and (2) Areas where secondary studies are needed to provide summarized empirical evidence to SWEBOK.

Search-based testing is an area with numerous secondary studies [S18, S20, S30, S46, S64, S65], which is missing from SWEBOK. Using clouding computing in testing with two secondary studies [S10, S42], and given its importance in the future it, should perhaps be included in (next versions of) SWEBOK. During recent years, the software industry has mostly transferred to Agile methods and Rapid Release cycles that puts special pressure on software testing. Thus, we suggest that the impact of such processes on software testing should be included in SWEBOK Chapter dealing with Software Testing [S26, SS49]. Currently, such things exits only in Chapter 9 (Section 4.4. Agile methods). Similarly we believe that Software Testing standards [S72] should be covered in SWEBOK in more depth.

We found that SWEBOK covers a larger set of topics than ISTQB foundation syllabus. This resulted in a higher number of topics that are not covered in the secondary studies. For details of missing (needed) secondary studies, we refer to Table 15. Next we highlight the areas which we consider the most important for improving the evidence of SWEBOK topics.

- From Section 1.2 of SWEBOK: We observed that "testability" is missing in the pool of secondary studies. Given the importance of testability, such a secondary study could be highly valuable also to industry.
- Section 2.2 of SWEBOK: Objectives of Testing have several sub-areas meriting secondary studies. For example, Alpha and Beta-testing are popular in the industry but even the primary studies of the topic are rare. A modern variant of Beta-testing namely A/B-testing could also be considered in that context. Furthermore, performance and stress testing is becoming more important as the majority of the systems run online and can exhibit large variation in the user-base.
- Similar to the ISTQB syllabus, SWEBOK also included experience- and intuition-based testing techniques or Exploratory Testing. This is an area that has no secondary studies. Such study could also include error-guessing method as it is often done based on experience and intuition.
- Similarly to ISTQB, SWEBOK has many test management topics residing under its Section 5 (Test Process). Again secondary studies could be made for example on the role of independence in software testing, test exit criteria used in the industry, and the role of product risk in test management.
- The existing secondary studies have mostly focused on test-case design, test generation and execution. This leaves the need for secondary studies in the areas of test-environment development and setup, test results evaluation and reporting.

7.1.3. Implications for the research community

Our tertiary study could provide various benefits and have impactful implications for the research community, several of which are discussed next. One could use our classification to create most popular secondary study target in terms academic and general popularity. In terms of academic popularity, measured by the number of secondary studies, the most popular secondary studied could have a title as follows: *Model-based Regression Testing of Web Services* which would combine the most popular method, target and phase (Model-based, Web-services, and Regression testing). In terms of general popularity of the title could be: *Testing Mobile applications with TDD at unit-testing phase* level combining the most popular part measured with Google hits (TDD, Mobile, and Unit testing). However, that title contains a conflict since TDD is really a development phase practice and when following TDD the unit testing phase is actually embedded in to the development phase.

Table 14
Mapping of items in ISTQB foundation syllabus to our results (items in the gray background highlight the open needs for secondary studies).

ISTQB foundation syllabus Headings	Sub-headings	Comment and references
1. Fundamentals of testing	1.1 Why is testing necessary?	We did not expect to find studies on this topic.
	1.2 What is testing?	
	1.3 Seven testing principles	
	1.4 Fundamental test process	Source [S85] reviewed 23 test process models majority based on TMMi or TPI
	1.5 The psychology of testing	
	1.6 Code of ethics	We did not expect to find studies on this topic.
2 Testing throughout the software life cycle	2.1 Software development models	One paper focused on software testing in Agile software development [S26] and other focused on Rapid Release cycle and software testing [S49].
	2.2 Test levels	Many papers focused on Unit-testing (or Component testing as it is called in ISTQB syllabus) [S1, S25, S39, S43, S98]. In the syllabus, TDD is part of Unit-testing which explains the difference between Section 6.1.3. Integration testing was dealt in three papers [S13, S68, S91] and Acceptance testing in one paper [S31]. System testing is often implicit, thus, saying what papers focus on system testing is difficult.
	2.3 Test types	Majority of papers focused on functional testing, but as with system testing this was often implicit. Testing of non-functional software characteristics was explicitly covered in two papers [S18, S73]. Regression testing was studied in 11 paper [S16, S23, S51, S53, S58, S59, S60, S61, S63 S76, S83]
	2.4 Maintenance testing	ISTQB considers maintenance testing, to be regression testing and new feature testing done for a new release of an existing product. Plenty of regression testing papers were found (see the above cell) but no paper on new feature testing or impact analysis for new releases. Three papers focused on test code engineering and test maintenance which can be considered as part of Maintenance Testing [S3, S67, S71]
3 Static techniques		ISTQB includes static techniques in their syllabus but such techniques were not part of our search scope. We found one source that considered how to combine static and dynamic quality assurance [S17].
4 Test design techniques	*4.1 The test development process*	No papers focused on the broad topic of the test development process. Rather the articles studied more focused individual techniques
	4.2 Categories of test design techniques	Two papers had studied and compared several test design techniques [S16, S63].
	4.3 Specification-based or black-box techniques	Several papers in this area focused on UML and model-based testing matching subsections 4.3.4 State Transition Testing and 4.3.5 Use Case Testing [S19, S22, S24, S35, S41, S44, S55, S68, S78, S82, S97]. One paper had focused on acceptance testing with decision tables [S31]. Additionally, the two studies focusing on several techniques provide relevant information for this topic [S16, S63]. Combinatorial testing had four papers [S4, S29, S30, S36] but to our surprise the topic is excluded from ISTQB syllabus.
	4.4 Structure-based or white-box techniques	The majority of the papers on white-box testing focused on Search-based testing where the goal is to automatically generate test data/input that covers 100% of the executions paths with the help of meta-heuristic search [S18 S20 S46 S63 S64 S65 S77]. To our surprise ISTQB does not mention Search-based testing at all. Furthermore, we found one paper studying coverage tools [S6]. Additionally, the two studies focusing on several techniques provide relevant information for this topic [S16, S63]. Two papers covered symbolic execution [S9, S30]
	4.5 Experience-based techniques	
	4.6 Choosing test techniques	No study had focused particularly on choosing test techniques. However, the two studies focusing on several techniques provide relevant information [S16, S63]. Also paper on to combining static and dynamic quality assurance techniques is relevant for this area [S17]. Additionally, papers focusing on particular SUT types, Section 6.1.2, may provide help in this area.
5. Test Management	*5.1 Test organization*	
	5.2. Test planning and estimation	No paper had focused on this topic. Paper studying the how different techniques can reduce test effort is partially relevant [S57]
	5.3 Test progress monitoring and control	
	5.4 Configuration management	
	5.5 Risk and testing	
	5.6 Incident management	Defect reporting that is part of software testing progress has been studied in two sources [79,80]. Yet, our search strings did not include these types of studies.
6. Tool support for testing	6.1. Types of Test Tools	Several papers reviewed different kind of tool support for testing [S1 S3 S6 S19 S29 S67 S71 S73 S98]. Those tools include tools for unit-testing [S1, S98], combinatorial testing [S29], and security [S98]
	6.2. Effective use of tools: potential benefits and risks	Some studies had also focused on the benefits and drawbacks of tool usage [S31,32,S34]
	6.3 Introducing a tool into an organization	

Gaps to be covered in future works were related to comparisons, “How does X differ from Y?”, relationships “Are X and Y related?”, causality, “Does X cause (or prevent) Y?”, and comparative causality, “Does X cause more Y than does Z?”. For each question type less than 5 RQs were available. Additionally, no RQs attempted to cover highly ambitious interaction for causality, “Does X or Z cause more Y under one condition but not others?” Thus, we suggest that researcher go boldly for the more ambitious relationship and causality questions to have deeper knowledge in the area of software testing. Perhaps, less emphasis should given for the more simple questions related to base rate. However, the selected RQs of secondary studies are largely affected by the primary studies, thus, such a change is also needed in the primary study level as well.

One could also speculate that perhaps the strict guidelines of SLRs and SMs wear out the writer and there is less intellectual effort put in such papers. Another reason could be that SMs and SLRs are often conducted with PhD students who are starting their careers and thus the output quality might not be high. It could also be that only individuals with enough academic successes and prestige are bold enough to do a regular surveys these days. Of course, those individuals often write high quality papers which lead to many citations. The presented reasons here are, of course, only speculations. Figuring out the reason for differences in citation behavior is an interesting future research topic.

Table 15
Mapped of Chapter 4 of the SWEBOK to our results (items in the gray background highlight the open needs for secondary studies).

Testing topics in SWEBOK Headings	Sub-headings	Comment and references
1. Software testing fundamentals	1.1. Testing-related terminology	We did not expect to find studies on this topic
	1.2. Key issues	The number of papers per key issues varied. Test Selection Criteria (Section 1.2.1) is implicitly covered in regression testing papers that try to select optimal test set [S16, S23, S51, S53, S58, S59, S60, S61, S63 S76, S83]. However, no study had this as their main topic. Testing effectiveness (1.2.2.) was studied in [S57] and the Oracle problem (1.2.4) in [S95].
		Yet there were areas with no papers namely: 1.2.3. Testing for Defect Identification, 1.2.5. Theoretical and Practical Limitations of Testing, 1.2.6. The Problem of Infeasible Paths, 1.2.7. Testability
	1.3. Relationship of testing to other activities	We found one source that considered how to combine static and dynamic quality assurance [S17], another one considered combining model-checkers and testing [S93], paper on combining formal verification and testing [S41].
		We found no papers comparing Testing with Debugging (1.3.3) and Programming (1.3.4.)
2. Test levels	2.1. The target of the test	Two papers focused on Unit-testing (2.1.1) [S1, S98]. Integration testing (2.1.2) was dealt in three papers [S13, S68, S91]. System testing (2.1.3) is often implicit, thus, saying what papers focus on system testing in particular is difficult.
	2.2. Objectives of testing	Most popular testing objective was regression testing (2.2.5) [S16, S23, S51, S53, S58, S59, S60, S61, S63 S76, S83]. Acceptance testing had one paper [S31] like Security testing [S73], and non-functional testing [S18].
		Areas with no studies were numerous: 2.2.2. Installation Testing 2.2.3. Alpha and Beta Testing, 2.2.6. Performance Testing, 2.2.8. Stress Testing, 2.2.9. Back-to-Back Testing, 2.2.10. Recovery Testing, 2.2.11. Interface Testing,
		2.2.13. Usability and Human Computer Interaction Testing was out of our scope as was
		2.2.4. Reliability Achievement and Evaluation
		2.2.12. Configuration Testing knowledge needs could be partially filled with numerous product-line testing papers: [S2, S7, S14, S50, S69, S70, S74, S79]
3. Test techniques	*3.1. Based on the software engineer's intuition and experience*	
	3.2. Input domain-based	Two studies focusing on several techniques provide relevant information for this topic [S16, S63]. Combinatorial testing had three papers [S4, S29, S36].
	3.3. Code-based techniques	The majority of the papers on white-box testing focused on Search-based testing where the goal is to automatically generate test data/input that covers 100% of the paths with meta-heuristic search [S18 S20 S46 S63 S64 S65 S77]. To our surprise, SWEBOK does not mention search-based testing at all. Furthermore, we found one paper studying coverage tools [S6].
		Additionally, the two studies focusing on several techniques provide relevant information for this topic [S16, S63]. Two papers covered symbolic execution [S9, S30].
	3.4. Fault-based techniques	Mutation testing was covered in several papers [S16, S21, S28, S48, S54, S63, S84].
		However, no papers exists for fault-guessing.
	3.5. Usage-based techniques	
	3.6. Model-based testing techniques	Several papers in this area focused on UML and model-based testing matching [S19, S22, S24, S35, S41, S44, S55, S68, S78, S82, S97]. One paper had focused on acceptance testing with decision tables [S31].
	3.7. Techniques based on the nature of the application	Several papers were found focusing on different application types, see Section 6.1.2
	3.8. Selecting and combining techniques	*A set of paper considered multiple techniques [S16, S22, S30, S41, S63], yet the practicalities of combination of the techniques were not studied.*
4. Test-related measures	4.1. Evaluation of the program under test	Our search string focused on testing and did not capture papers e.g. on reliability growth models [83], failure models [84], and other means of evaluating SUT. Thus, we cannot comment whether evidence is sufficient for this area.
	4.2. Evaluation of the tests performed	The majority of the papers listed for SWEBOK Section 3 can also be listed here. Search-based testing [S18 S20 S46 S63 S64 S65 S77], symbolic execution [S9, S30], mutation [S16, S21, S28, S48, S54, S63, S84], comparison and effectiveness of different techniques [S16, S17, S57, S63].
5. Test process	5.1. Practical considerations	Source [S85] reviewed 23 test process models majority based on TMMi or TPI. Additionally, one paper focused on software testing in Agile software development [S26] and other focused on Rapid Release cycle and software testing [S49]. Most frequently studied process area of SWEBOK has been Test-Driven Development with 3 papers [S25, S39,S43] *Areas with no studies were numerous 5.1.2. Test Guides 5.1.4. Test Documentation and Work Products 5.1.6. Internal vs. Independent Test Team 5.1.7. Cost/Effort Estimation and Test Process Measures 5.1.8. Termination.*
	5.2. Test activities	Test-case generation and execution had numerous papers as all papers listed in Section 6.1.1 focused on one or both of those. Problem reporting and defect tracking are not captured by our search strings but evidence on them exits as well sources [79,80].
		However, no papers were found on topics 5.2.1. Planning 5.2.3. Test Environment Development 5.2.5. Test Results Evaluation
6. Software testing tools	6.1. Testing tool support	Some studies had also focused on the benefits and drawbacks of tool usage [S31, S32, S34]
	6.2. Categories of tools	Several papers reviewed different kind of tool support for testing [S1, S3, S6, S19, S29, S67, S71, S73, S98]. Those tools include tools for unit-testing [S1, S98], combinatorial testing [S29], and security [S98]

7.2. Threats to validity

According to Petersen et al., similar to empirical studies, validity considerations are also applicable to SM and SLR studies [16,30]. Similar to how we considered and mitigated the potential threats to validity in our previous SM and SLR studies [17–20,23–26], we considered and mitigated them as follows.

The main issues related to threats to validity of this tertiary study are inaccuracy of data extraction, and incomplete set of studies in our pool due to limitation of search terms, selection of academic search engines, and researcher bias with regards to exclusion/inclusion criteria. In this section, these threats are discussed in the context of the four types of threats to validity based on a standard checklist for validity threats presented in [85].

7.2.1. Internal validity

The systematic approach that has been utilized for article selection is described in Section 4. In order to make sure that this review is repeatable, search engines, search terms and inclusion/exclusion criteria are carefully defined and reported. Problem-

atic issues in selection process are limitation of search terms and search engines, and bias in applying exclusion/inclusion criteria.

Limitation of search terms and search engines can lead to incomplete set of primary sources. Different terms have been used by the author to point to a similar concept. In order to mitigate risk of finding all relevant studies, formal searching using defined keywords has been done followed by manual search in references of initial pool and in web pages of active researchers in our field of study. For controlling threats due to search engines, we have included comprehensive academic databases such as Google Scholar. Therefore, we believe that adequate and inclusive basis has been collected for this study and if there is any missing publication, the rate will be negligible.

Applying inclusion/exclusion criteria can suffer from researchers' judgment and experience. Personal bias could be introduced during this process. To minimize this type of bias, the authors used a joint voting mechanism to include/exclude papers in the pool. Also, the authors' expertise and experience in various recent SM and SLR studies [17–20,23–26] have been very helpful in this aspect.

7.2.2. Construct validity

Construct validities are concerned with issues that to what extent the object of study truly represents theory behind the study [85]. Threats related to this type of validity in this study were suitability of RQs and categorization scheme used for the data extraction.

To limit construct threats in this study, the GQM approach is used to preserve the tractability between research goal and questions. Research questions are designed to cover our goal and different aspects of secondary studies in the area of software testing. Questions are answered according to a categorization scheme. For designing a good categorization scheme, we have adapted standard classifications from [16] and also have finalized the used schema through several iterative improvement process.

7.2.3. Conclusion validity

Conclusion validity of a SM study deals with whether correct conclusions are reached through rigorous and repeatable treatment. In order to ensure reliability of our treatments, the terminology of the defined schema was reviewed and iteratively improved by checking it against the standard terminology in software testing books. All the secondary studies were reviewed by one author and the extracted data were peer reviewed by the other author to mitigate bias in data extraction. Each disagreement was resolved by discussions and consensus among researchers.

Following the systematic approach and described procedure ensured replicability of this study and assured that results of similar study will not have major deviations from our classification decisions.

7.2.4. External validity

External validity is concerned with to what extent the results of this tertiary study can be generalized. As described in Section 4, defined search terms in the article selection approach resulted in having all secondary studies 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 secondary studies in software testing. For these issues, we argue that relevant literature we selected in our pool contained sufficient information to represent the knowledge reported by previous researchers or professionals.

Also, note that our findings in this study are mainly within the field of secondary studies in software testing. Beyond this field, we had no intention to generalize our results. Therefore, few problems with external validity are worthy of substantial attention.

8. Conclusions and future work

The goal of this tertiary study was to systematically map (classify) the state-of-the-art in secondary studies in the area of software testing, to find out the recent trends and directions in this field, and to identify opportunities for future research, from the point of view of researchers and practitioners in this area.

Our study pool included a set of 101 secondary studies published in the area of software testing between 1994 and 2015. Our mapping data is available through an online publicly-accessible repository. The research volumes of knowledge in software testing is huge. Any newcomer or industrial participant is likely to experience difficulties in addressing it. Thus, we propose that tertiary studies can act as summarizing indexes to those people easing the way to find the most relevant information. Additionally, we suggest that tertiary studies can help in content selection for university level courses as ideally the contents should be based on the data (popularity of a particular area) rather than sole discretion of the course teacher. For example the second author has already used this study for such purpose when selecting teaching topics at the University of Oulu.

Among our detailed findings are the following: (1) Model-based approach is the most popular testing method, web services are the most popular type of system under test (SUT), while regression testing is the most popular testing phase in terms of number of secondary studies. (2) When looking at popularity in terms Google hits we found Model-based testing is the most popular method, mobile application are the most popular target, and unit-testing is the most popular phase. (3) There is a need for more secondary studies in performance, load, mobile, non-automated (manual), security testing and TDD. (4) Albeit the systematic nature of SM and SLR studies, regular surveys are still being conducted and published. (4) The quality secondary studies is slowly increasing as the years go by. (5) We found that on average secondary studies receive more citations, however, the difference was not statistically significant. (6) We also found that regular surveys receive significantly more citations than SMs or SLRs but the reasons for this are unclear and need future studies. and (7) The secondary study with the highest number of citations (1052 times) is a survey on methods of testing finite state machines, and has been published in 1996.

There are several future work directions after this tertiary study. For example, similar to a tertiary study in general SE [8], we intend to conduct critical appraisal of SMs and SLRs in the software testing and other areas from the perspective of various aspects of SMs and SLRs, e.g., research questions. Also this tertiary study can be conducted in other sub-areas of SE, e.g., software maintenance and software requirements. We plan to conduct a separate tertiary study to synthesize all that empirical evidence.

We also plan to extend and conduct further empirical cases to assess the benefits of this tertiary study to (young) researchers, e.g., PhD students, and practitioners. For example, we plan to conduct a survey study with our industry partners by providing to them the list of RQs in the secondary studies in a sub-area, e.g., web app testing, and asking them to evaluate the usefulness of the RQs which will help assess the "index" metaphor (this study acting as an index to the software testing field). Such an assessment will be quite similar to studies on practitioners' perception of the relevance of software engineering research, e.g., [86].

To extend our initial analysis in RQ4, we plan to extend the analysis by running correlations between number of citations w.r.t. number of primary studies and quality measures of secondary studies. We may even be able to build a regression model to predict citations to secondary studies.

Acknowledgements

Vahid Garousi was partially supported by several internal grants by Hacettepe University and the Scientific and Technological Research Council of Turkey (TÜBİTAK) via grant #115E805.

References

[1] J. Jill, M. Lydia, L. Fiona, Doing Your Literature Review: Traditional and Systematic Techniques, SAGE Publications, 2011.
[2] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, EBSE Technical Report EBSE-2007-01, Software Engineering Group, Keele University, UK, 2007.
[3] Evidence-Based Software. Engineering (EBSE), Software engineering evidence map. http://www.dur.ac.uk/ebse/evidence.php. Last Accessed: July.
[4] B. Kitchenham, P.B. Keele, D. Budgen, The educational value of mapping studies of software engineering literature, in: Proceedings of ACM/IEEE International Conference on Software Engineering, 2010, pp. 589–598.
[5] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, Using mapping studies in software engineering, Evidence-Based Software Engineering, 2008.
[6] B. Kitchenham, O.P. Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Inf. Softw. Technol. 51 (2009) 7–15.
[7] B. Kitchenham, R. Pretorius, D. Budgen, O. Pearl Brereton, M. Turner, M. Niazi, et al., Systematic literature reviews in software engineering – a tertiary study, Inf. Softw. Technol. 52 (2010) 792–805.
[8] F.Q.B.d. Silva, A.L.M. Santos, S.C.B. Soares, A.C.C. França, C.V.F. Monteiro, A critical appraisal of systematic reviews in software engineering from the perspective of the research questions asked in the reviews, in: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2010.
[9] D.S. Cruzes, T. Dyba, Research synthesis in software engineering: A tertiary study, Inf. Softw. Technol. 53 (2011) 440–455.
[10] F.Q.B. da Silva, A.L.M. Santos, S. Soares, A.C.C. França, C.V.F. Monteiro, F.F. Maciel, Six years of systematic literature reviews in software engineering: An updated tertiary study, Inf. Softw. Technol. 53 (2011) 899–913.
[11] G.K. Hanssen, D. Smite, N.B. Moe, Signs of agile trends in global software engineering research: a tertiary study, in: International Conference on Global Software Engineering Workshop, 2011, pp. 17–23.
[12] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, Inf. Softw. Technol. 53 (2011) 625–637.
[13] A.B. Marques, R. Rodrigues, T. Conte, Systematic literature reviews in distributed software development: a tertiary study, in: International Conference on Global Software Engineering, 2012, pp. 134–143.
[14] S. Imtiaz, M. Bano, N. Ikram, M. Niazi, A tertiary study: experiences of conducting systematic literature reviews in software engineering, in: Presented at the Proceedings of International Conference on Evaluation and Assessment in Software Engineering, 2013.
[15] J.M. Verner, O.P. Brereton, B.A. Kitchenham, M. Turner, M. Niazi, Risks and risk mitigation in global software development: A tertiary study, Inf. Softw. Technol. 56 (1) (2014) 54–78.
[16] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, 12th International Conference on Evaluation and Assessment in Software Engineering (EASE) Presented at the, 2008.
[17] I. Banerjee, B. Nguyen, V. Garousi, A. Memon, Graphical user interface (GUI) testing: systematic mapping and repository, Inf. Softw. Technol. 55 (2013) 1679–1694.
[18] V. Garousi, A. Mesbah, A. Betin-Can, S. Mirshokraie, A Systematic mapping study of web application testing, Elsevier J. Inf. Softw. Technol. 55 (2013) 1374–1396.
[19] S. Doğan, A. Betin-Can, V. Garousi, Web application testing: a systematic literature review, J. Syst. Softw. 91 (2014) 174–201.
[20] V.G. Yusifoğlu, Y. Amannejad, A. Betin-Can, Software test-code engineering: a systematic mapping, J. Inf. Softw. Technol. 58 (2014) 123–147.
[21] 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.
[22] M. Mäntylä, B. Adams, F. Khomh, E. Engström, K. Petersen, On rapid releases and software testing: a case study and a semi-systematic literature review, Empirical Softw. Eng. 20 (2015) 1384–1425 2015/10/01.
[23] R. Farhoodi, V. Garousi, D. Pfahl, J.P. Sillito, Development of scientific software: a systematic mapping, bibliometrics study and a paper repository, Int. J. Softw. Eng. Knowl. Eng. 23 (4) (2013) 463–506.
[24] V. Garousi, Classification and trend analysis of UML books (1997-2009), J. Softw. Syst. Model. (SoSyM) 11 (2) (2012) 273–285.
[25] V. Garousi, S. Shahnewaz, D. Krishnamurthy, UML-driven software performance engineering: a systematic mapping and trend analysis, in: V.G. Díaz, J.M.C. Lovelle, B.C.P. García-Bustelo, O.S. Martínez (Eds.), Progressions and Innovations in Model-Driven Software Engineering, IGI Global, 2013.
[26] J. Zhi, V.G. Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, G. Ruhe, Cost, benefits and quality of software development documentation: a systematic mapping, J. Syst. Softw. 99 (2015) 175–198.
[27] J. VANHANEN, M.V. MÄNTYLÄ, A systematic mapping study of empirical studies on the use of pair programming in industry, Int. J. Softw. Eng. Knowl. Eng. 23 (2013) 1221–1267.
[28] J. Zhi, V.G. Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, G. Ruhe, Cost, benefits and quality of software development documentation: a systematic mapping, J. Syst. Softw. 99 (2015) 175–198.
[29] E. Kupiainen, M.V. Mäntylä, J. Itkonen, Using metrics in agile and lean software development – a systematic literature review of industrial studies, Inf. Softw. Technol. 62 (6// 2015) 143–163.
[30] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: an update, Inf. Softw. Technol. 64 (2015) 1-18, 8.
[31] 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.
[32] F. Elberzhager, J. Münch, V.T.N. Nha, A systematic mapping study on the combination of static and dynamic quality assurance techniques, Inf. Softw. Technol. 54 (2012) 1–15.
[33] V.R. Basili, Software modeling and measurement: the Goal/Question/Metric paradigm, University of Maryland at College Park, 1992 Technical Report.
[34] S. Easterbrook, J. Singer, M.-A. Storey, D. Damian, Selecting empirical methods for software engineering research, in: F. Shull, J. Singer, D.K. Sjøberg (Eds.), Guide to Advanced Empirical Software Engineering, Springer, London, 2008, pp. 285–311.
[35] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, London, England, United Kingdom, 2014.
[36] Centre for Reviews and Dissemination, What are the criteria for the inclusion of reviews http://www.york.ac.uk/inst/crd/faq4.htm, 2007.
[37] R.V. Binder, Testing object-oriented software: a survey, in: Proceedings of the Tools-23: Technology of Object-Oriented Languages and Systems, 1996, p. 374.
[38] R. Lai, A survey of communication protocol testing, J. Syst. Softw. 62 (2002) 21–46.
[39] N. Juristo, A.M. Moreno, S. Vegas, Reviewing 25 years of testing technique experiments, Empirical Softw. Eng. 9 (2004) 7–44.
[40] P. McMinn, Search-based software test data generation: a survey, Softw. Test. Verification Reliab. 14 (2004) 105–156.
[41] M. Grindal, J. Offutt, S.F. Andler, Combination testing strategies: a survey, Softw. Test. Verification Reliab. 15 (2005).
[42] L.M. Karg, M. Grottke, A. Beckhaus, A systematic literature review of software quality cost research, J. Syst. Softw. 84 (3// 2011) 415–427.
[43] V. Garousi, Online paper repository for systematic mapping of secondary studies in software testing, http://goo.gl/Oxb0X8, Last accessed: July.
[44] R. Kumar, K. Singh, A literature survey on black-box testing in componen-t-based software engineering, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 2 (2012) 420–423.
[45] S. Nagpurkar, Y.B. Gurav, A Survey on test case generation from UML based requirement analysis model, Int. J. Advancements Res. Technol. 2 (2013).
[46] N. Juristo, A.M. Moreno, S. Vegas, A survey on testing technique empirical studies: how limited is our knowledge, in: Proceedings of International Symposium in Empirical Software Engineering, 2002, pp. 161–172.
[47] R. Dorofeeva, K. El-Fakih, S. Maag, A.R. Cavalli, and N. Yevtushenko, "FSM-based conformance testing methods: a survey annotated with experimental evaluation," vol. 52, pp. 1286–1297, 2010.
[48] V. Garousi, M.V. Mäntylä, Bibliographic details for the repository of secondary studies in software testing. https://goo.gl/nCwvkm. Last accessed: Dec.
[49] 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.
[50] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Inf. Softw. Technol. 51 (2009) 957–976.
[51] 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.
[52] P. McMinn, Search-based software testing: past, present and future, in: IEEE International Conference on Software Testing, Verification and Validation Workshops, 2011, pp. 153–163.
[53] A.T. Endo, A.d.S. Simao, A systematic review on formal testing approaches for web services, in: Brazilian Workshop on Systematic and Automated Software Testing, International Conference on Testing Software and Systems, 2010, p. 89.
[54] A. Sharma, T.D. Hellmann, F. Maurer, Testing of web services – a systematic mapping, in: Proceedings of the IEEE World Congress on SERVICES, 2012, pp. 346–352.
[55] M. Bozkurt, M. Harman, Y. Hassoun, Testing Web Services: A Survey, Department of Computer Science, King's College London, 2010 Technical Report TR-10-01.
[56] D.S. Cruzes, T. Dyba, Recommended steps for thematic synthesis in software engineering, in: International Symposium on Empirical Software Engineering and Measurement (ESEM), 2011, pp. 275–284.
[57] M.B. Miles, M. Huberman, Qualitative Data Analysis: An Expanded Sourcebook, SAGE Publications, 1994.
[58] HP Capgemini, Sogetti, World quality report 2014–2015. http://www.sogeti.com/solutions/testing/wqr/, Last accessed: Sept.
[59] G. Fraser, A. Arcuri, A large-scale evaluation of automated unit test generation using EvoSuite, ACM Trans. Softw. Eng. Methodol. 24 (2014) 1–42.

[60] Y. Singh, A. Kaur, B. Suri, S. Singhal, Systematic literature review on regression test prioritization techniques, Informatica 36 (2012).
[61] 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 Software Engineering Languages and Technologies, 2007.
[62] D. Lee, M. Yannakakis, Principles and methods of testing finite state machines-a survey, in: Proceedings of the IEEE, 84, 1996, pp. 1090–1123.
[63] 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.
[64] B. Haugset, G.K. Hanssen, Automated acceptance testing-a literature review and an industrial case study, in: Agile Conference, 2008, pp. 27–38.
[65] V. Garousi, J.M. Fernandes, Highly-cited papers in software engineering: the top-100, Inf. Softw. Technol. 71 (2016) 108–128.
[66] C. Wohlin, An analysis of the most cited articles in software engineering journals – 2001, Inf. Softw. Technol. 50 (2008) 3–9.
[67] C. Wohlin, An analysis of the most cited articles in software engineering journals – 2002, Inf. Softw. Technol. 51 (2009) 2–6.
[68] D.W. Aksnes, Characteristics of highly cited papers, Res. Eval. 12 (2003) 159–170.
[69] R.V. Noorden, B. Maher, R. Nuzzo, The top 100 papers, Nature 514 (2014) 550–553.
[70] J. Antonakis, N. Bastardoz, Y.H. Liu, C.A. Schriesheim, What makes articles highly cited? Leadership Q. 25 (2014) 152–179.
[71] V. Garousi, M.V. Mäntylä, in: Citations, Research Topics and Active Countries in Software Engineering: A Bibliometrics Study, vol. 19, Elsevier Computer Science Review, 2016, pp. 56–77.
[72] V. Garousi, A bibliometric analysis of the Turkish software engineering research community, Springer J. Scientometrics 105 (2015) 23–49.
[73] V. Garousi, G. Ruhe, A bibliometric/geographic assessment of 40 years of software engineering research (1969-2009), Int. J. Softw. Eng. Knowl. Eng. 23 (2013) 1343–1366.
[74] M. Niazi, Do systematic literature reviews outperform informal literature reviews in the software engineering domain? An initial case study, Arabian J. Sci. Eng. 40 (2015) 845–855 2015/03/01.
[75] ISTQB (International Software Testing Qualifications Board), Certified tester foundation level syllabus, http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html. 2011
[76] ISTQB (International Software Testing Qualifications Board), ISTQB® Worldwide Software Testing Practices Report (2015-2016), 2016 http://www.istqb.org/references/surveys/istqb-worldwide-software-testing-practices-report.html.
[77] ISTQB (International Software Testing Qualifications Board), ISTQB portfolio, http://www.istqb.org/portfolio_map/images/istqb-portfolio-image.jpg. Last accessed: Aug 2016.
[78] D. Radjenović, M. Heričko, R. Torkar, A. Živkovič, Software fault prediction metrics: a systematic literature review, Inf. Softw. Technol. 55 (8) (2013) 1397–1418.
[79] J.D. Strate, P.A. Laplante, A literature review of research in software defect reporting, IEEE Trans. Reliab. 62 (2013) 444–454.
[80] Y.C. Cavalcanti, P.A. da Mota Silveira Neto, I.d.C. Machado, T.F. Vale, E.S. de Almeida, S.R.d.L. Meira, Challenges and opportunities for software change request repositories: a systematic mapping study, J. Softw. 26 (2014) 620–653.
[81] P. Bourque, R.E. Fairley, Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0, IEEE Computer Society Press, 2014.
[82] ACM, Graduate software engineering 2009 (GSwE2009): Curriculum guidelines for graduate degree programs in software engineering, http://www.acm.org/education/curricula-recommendations.
[83] F. Febrero, C. Calero, M.Á. Moraga, A systematic mapping study of software reliability modeling, Inf. Softw. Technol. 56 (8// 2014) 839–849.
[84] L. Feinbube, P. Tröger, A. Polze, 2016 The landscape of software failure cause models, http://arxiv.org/abs/1603.04335.
[85] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, 2000.
[86] D. Lo, N. Nagappan, T. Zimmermann, How practitioners perceive the relevance of software engineering research, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, Bergamo, Italy, 2015.