Received: 3 April 2015 | Revised: 4 February 2017 | Accepted: 14 May 2017

DOI: 10.1002/stvr.1639

WILEY

RESEARCH ARTICLE

Impediments for software test automation: A systematic literature review

Kristian Wiklund[1,2] | Sigrid Eldh[1,2] | Daniel Sundmark[2] | Kristina Lundqvist[2]

[1]Ericsson AB, Stockholm, SE-164 80, Sweden
[2]Mälardalen University, Box 883, Västerås, SE-721 23, Sweden

Correspondence
Daniel Sundmark, Mälardalen University, Box 883, SE-721 23, Västerås, Sweden.
Email: daniel.sundmark@mdh.se

Funding information
Ericsson AB; Mälardalen University; Knowledge Foundation

Summary

Automated software testing is a critical enabler for modern software development, where rapid feedback on the product quality is expected. To make the testing work well, it is of high importance that impediments related to test automation are prevented and removed quickly. An enabling factor for all types of improvement is to understand the nature of what is to be improved. We have performed a systematic literature review of reported impediments related to software test automation to contribute to this understanding. In this paper, we present the results from the systematic literature review: The list of identified publications, a categorization of identified impediments, and a qualitative discussion of the impediments proposing a socio-technical system model of the use and implementation of test automation.

KEYWORDS

impediments, software testing, systematic literature review, test automation, test execution

1 | INTRODUCTION

Software testing is by far the most commonly used method for quality assurance and quality control in a software development organization, and a very important part of the development process [IP1]. Testing makes it possible to get information about the difference between the actual and the required behavior of the software product [1] when delivered. The importance and complexity of software testing is reflected by the costs involved: 30% to 80% of the development costs are reported to be related to testing [2,3][IP2][4], and studies on release time indicate that most of the release time is consumed by testing [5]. The cost and time involved in testing can be managed through test automation, where the execution of a test is performed by software instead of a human.

Test automation can be used to improve the process effectiveness, for example, by reducing the risk for human errors [IP3] and make the tests more repeatable [6]. Test automation can also be used to improve the process efficiency, for example, by enabling the use of the staff for other tasks [IP4] and perform more testing in less time [6][IP5]. Test automation is a core component in agile development [IP6], where it is used both for rapid feedback and to enable testing by everyone that deliver code. Methods such as continuous integration [7], test-driven development, and automated acceptance testing [8] makes test automation an everyday activity in an agile organization [9]. The high use of test automation also makes the automated tests mission critical: If the test automation stops working or slows down, the development will also stop or slow down.

The investment cost for test automation can be significant. It is difficult to propose a generalizable cost model, due to a lack of published information available on the cost of ownership for test automation. However, the size and complexity of a test system can be in the order of, or often larger than, the complexity and size of the tested product [10,11][IP7][12]. This makes it reasonable to conjecture that the cost of ownership of an automated test execution system is in the order of the cost of development of the product it is testing. Even if a lot less effort is spent on the test systems than on the sold products, it still represents a major investment for the software business as a whole. If the test automation does not perform as well as expected, this investment is unlikely to be recovered as planned, which may have significant consequences for the business. Further, it is reasonable to conjecture that test systems suffer from similar *technical debt* issues as regular systems do. Technical debt is a term that is used to describe the gap between the current state of the system and the ideal state of the system [13], usually in situations where a compromise is made during development to meet demands in one dimension, such as lead time, by sacrificing work in another dimension, such as architecture.

In a previous case study in the telecommunications sector, we investigated issues related to test automation by analyzing the contents of an online discussion board [IP8]. In this study, we extend the context beyond the studied case and focus on more generally investigating impediments for

industrial test automation. Impediments are defined as "anything that prevents a team member from performing work as efficiently as possible" [14]. It should also be noted that the review focuses on industrial studies of test automation. Thereby, it does not consider the large body of knowledge on methods for automated test generation, an area where recent and mature reviews already exist (see, eg, Anand et al [15]).

From industrial experience, available publications in the field, and our own research, we propose that many impediments experienced by software developers are related to the use of automated testing. Many test automation impediments appear to be of a general nature, as they are present in reports from widely different parts of the software industry. Considering the importance of testing in software development, test automation in agile development [IP6], and the large costs involved, we propose that there is a general need for systematic knowledge about what impediments could be encountered and reasons that the impediments occur. Knowledge about the impediments is important to be able to reduce the impact of the impediments, or eliminate the impediments completely.

This paper contributes the following: (1) A systematic literature review (SLR) of research publications identifying real-life experiences of impediments related to software test automation, (2) a categorization of the identified impediments, (3) a model presenting how different phenomena encountered in test automation use and development interact to amplify the impediments, (4) a qualitative discussion on the impediment and phenomena, and (5) a proposal for future work.

The paper is organized as follows: An overview of the study design is presented in Section 2, followed by the results of the SLR together with the qualitative synthesis in Section 3. Section 5 discusses implications for practitioners and researchers, together with proposals for future work. Finally, the paper is concluded in Section 6.

2 | STUDY DESIGN

The design of the study draws primarily from two sources. The searching and screening approach largely follows the systematic literature review guidelines by Kitchenham et al. [16] and the synthesis is influenced by Noblit and Hare's meta-ethnographic method for qualitative synthesis [17].

2.1 | Objectives

There are two objectives for this research:

1. To identify scientifically reported software development impediments related to automated software testing.
2. To explain the mechanisms leading to the impediments through a qualitative synthesis.

2.2 | Process

To analyze the included publications and perform the synthesis the following process was used:

1. **Getting started**
   The meta-ethnographic method [17] defines "getting started" as determining what to be investigated using the method. This is formulated as two objectives, as presented in Section 2.1.
2. **Deciding what is relevant to the initial interest**
   In this phase, we define the search terms, the databases to be searched, the inclusion and exclusion criteria, and how to do the screening. A pilot screening activity was performed to calibrate the screening process by comparing differences between screening decisions taken by the different researches. Then, we performed the searches and the screening, finally resulting in the primary studies listed in Table 7 and in the "Included Publications" section at the end of this paper. The search strategy is described in Section 2.3, and the screening process is described in Section 2.5.
3. **Reading the studies**
   The purpose of this step is to get as familiar as possible with the content of the included studies [18] and extract the initial themes and concepts. The step can be viewed as preprocessing for the synthesis step.
4. **Synthesising the results**
   The synthesis phase contains the main difference between our study design and the original meta-ethnographic approach. Noblit and Hare propose that studies shall be translated into each other to identify common concepts, to form either reciprocal, refutational, or line of argument relations [17]. The relations are interpreted by the researcher, who expresses the synthesis in writing. This analysis becomes very complex as the number of studies grow. Atkins et al. [18] analysed approximately the same number of studies (n = 44) as in our study and concluded that it most likely would be impossible to translate the studies into each other. To manage this situation, they elected to perform a thematic analysis of the data instead of formally mapping the studies onto each other. Since we had a similar volume of studies, we elected to use the thematic method as well, as described by Cruzes and Dybå [19]. The generated themes and concepts were then analyzed in line with Noblit and Hare's original intent.

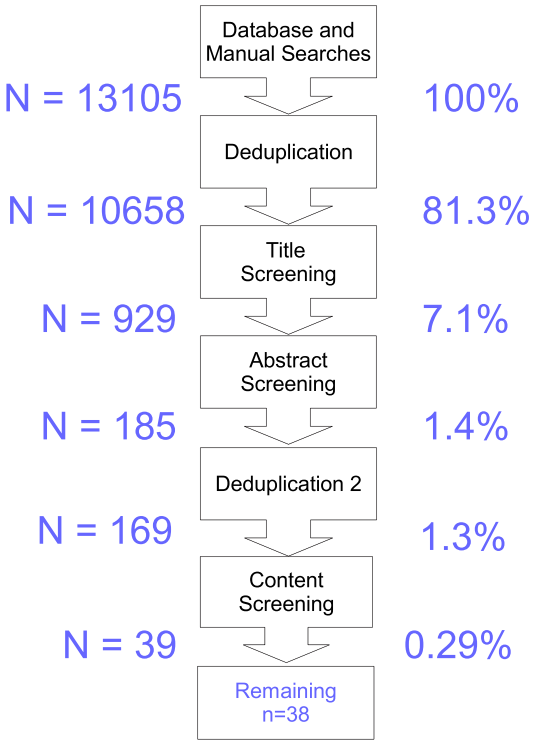

FIGURE 1 An overview of the screening process, showing the reduction in included studies per step

2.3 | Study retrieval

Failing to retrieve all relevant reports, the gold standard, is a significant threat to validity in literature reviews [20,21]. The "gold standard" for a literature search is the set of all reports that are relevant for the defined research objectives [22]. If the search strategy is perfectly adapted to the objectives, all relevant reports would be retrieved and all irrelevant reports would be ignored. For reasons described below, it is very difficult or impossible to retrieve the gold standard in a software engineering literature search [22].

Recall is the percentage of all relevant document in the searched databases that have been retrieved, and the precision is the percentage of retrieved documents that are relevant [20]. This is also known as the sensitivity of the search [22].

The terminology used in software engineering in general and testing in particular is not clearly defined [4,23–28]. The lack of a clearly defined terminology have at least two implications for a systematic literature review: The search terms have to be more inclusive to minimize the risk of omitting an important publication and will return more publications due to matching out of scope publications using the same terminology for a different concept. Publication keywords can be used to narrow the search further than a full-text or abstract search. However, the keywords suffer from the same issues as the terminology in general and are not always relevant for systematic literature reviews [28]. The consequence of the ambiguous terminology is a search with high recall and low precision that has to be refined by a series of screening actions as described in Section 2.5 and shown in Figure 1. The screening removes the irrelevant publications and produces the final set of studies for analysis.

2.3.1 | Database selection

In searching for relevant evidence, we decided to use the IEEE Xplore, Scopus, and Web of Science publication databases. It should be noted that this selection does not exclude ACM journals, such as Transactions on Software Engineering and Methodology (TOSEM). These publications are indexed by the Science Citation Index, which is included in the Web of Science database. Further, the Science Citation Index also indexes relevant Wiley and Springer journals, such as "Software Testing, Verification & Reliability," while Elsevier journals and proceedings, such as "Lecture Notes in Computer Science," are included in the Scopus database.

2.3.2 | Search query

The primary search query is shown in Table 1 using the PICO format [29]. In addition to this, we limited the searches by keywords and areas to keep the volume manageable, as specified for each database in Table 2.

2.4 | Validation of the searches

To validate the search query and database selection, we used a number of preidentified relevant publications, listed in Table 3. This revealed that Karl- [illegible] missing from the database search result. The reason is likely that it is an improvement-oriented paper and presents a process

TABLE 1 Search terms for the research objectives

Row	Terms
(P)	software AND (empirical OR industrial OR practical OR report OR case study OR survey OR experience*)
(I)	test* OR validation OR verification
(C)	
(O)	automat* OR tool* OR technical debt OR impediment* OR success critical factor*

The complete query is formulated by combining the rows with AND.

TABLE 2 Keywords and search area restrictions per database

Database	Additional database-specific restrictions
Scopus	subject area:computer science AND (keyword:software engineering OR
	keyword:software testing OR keyword:testing OR keyword:verification OR
	keyword:computer aided software engineering OR automation)
WoS	(NOT subject area:arts and humanities AND NOT subject area:social sciences)
	AND subject area:computer science
IEEE Xplore	subject area:computing & processing (hardware/software)

TABLE 3 Publications used for search validation

[IP4]	Establishment of Automated Regression Testing at ABB: Industrial Experience Report on 'Avoiding the Pitfalls,' Persson, C. and Yilmazturk, N. (2004)
[IP9]	A Minimal Test Practice Framework for Emerging Software Organizations, Karlström, D., Runeson, P., and Nordén, S. (2005)
[IP10]	Observations and Lessons Learned from Automated Testing, Berner, S., Weber, R., and Keller, R.K. (2005)
[IP3]	Empirical Observations on Software Testing Automation., Karhu K, Repo T, TaipaleO, and Smolander K. (2009)
[IP7]	Software Test Automation in Practice: Empirical Observations, Kasurinen J, TaipaleO, and Smolander K. (2010)
[IP11]	Technical Debt in Test Automation, Wiklund K, Eldh S, Sundmark D, and Lundqvist K. (2012)

improvement framework, based on empirical data, together with the empirical evaluation of the framework. The empirical study is presented in the paper, but it is not the main focus of the report.

As a consequence, we decided to perform backward and forward chaining [20] from the primary studies identified in the database searches [30], and an additional search of four journals [20] relevant for the research objectives.

The "IBM Journal of Research and Development" and the "IBM Systems Journal" were searched using the IBM search engine for papers matching "test," "validation," "verification," or "automation/automated." The remaining search criteria were applied manually after retrieval of a list of inclusion candidates.

The Software testing, Verification & Reliability journal and the Advances in Software Engineering journal were hand searched for papers matching the inclusion criteria. Papers found in these searches, as well as those identified by backward and forward chaining, were added to the database and were subsequently subjected the same screening process as the papers found by the regular search string.

2.5 | Study selection process

The study is limited to journal, conference, and magazine publications written in English. The inclusion and exclusion criteria for each phase in the review are described further below.

To increase the precision [20] of the search, a series of screenings were performed, as illustrated by Figure 1. Following the recommendations in the Cochrane Handbook for Systematic Reviews of Interventions [29, p. 179], we have not calculated Cohen's Kappa [31] to assess the interrater reliability. Instead, we performed a pilot screening and focused on a qualitative discussion of the synthesis and included items.

The screening process followed the steps described below:

- **Merge search results and remove duplicate records of the same report**
  As shown in Figure 1, there was a relatively large overlap between databases, which made automation in the shape of a Unix script necessary. The automated script removed duplicates in a two-step process. The first step identified duplicates by comparing the DOI of the publications, and the second step identified duplicates in publications missing DOI information by comparing the abstracts. If 2 abstracts were nonempty and byte-identical, the publications were considered to be duplicates.
- **Examine titles to remove obviously irrelevant reports**
  We embraced the Cochrane handbook recommendation to be over-inclusive [29] in this stage.
  Exclusion criteria: Obviously out-of-domain papers, eg, medicine, test of noncomputerized systems such as turbines or hydraulics, circuit design, [illegible], automatic test case generation, automatic test suite management, and editorials.

TABLE 4 Quality criteria and results

QC1	**Is the study based on empirical research or observation in industry?**
QC2	Are the research questions or objectives clearly stated?
QC3	Is there a description of the object of study and its context?
QC4	Is there a systematic research method defined prior to the study?
QC5	Is there a clear statement of the findings?

- **Examine abstracts to remove obviously irrelevant reports**
  *Inclusion criteria:*Software development in the real world, papers that may be within scope and cannot be excluded with good confidence.
  Exclusion criteria: Out-of-domain papers.
- **Manual deduplication**
  During abstract screening, we realized that there were duplicates remaining that had escaped the automatic deduplication.
  *Exclusion criteria:*Identical papers. We sorted the titles and inspected papers with the same title, then duplicates were removed.
- **Content screening: introduction, conclusions, and full text**
  - *Inclusion criteria:* Real-world experiences of technical debt and impediments related to software test automation and software test execution tools, as well as literature reviews and other secondary studies of the same. We started by analyzing the introduction and conclusions of the candidate reports. If this did not provide sufficient information, we continued immediately by analysing the rest of the report for inclusion or exclusion. The screening continued during detailed reading and initial theme extraction, when the question "Is this relevant to the research objectives and synthesis topic" was asked [32]. If the report was found to be irrelevant during thematic analysis, by not containing any relevant themes, it was excluded from the review.
  - *Exclusion criteria:* Fixed-design surveys that only provide tool information from closed-ended response [33] questions. The reason for this exclusion criteria is that a closed-ended question make a priori assumptions on what the impediments are, which results in a researcher bias [34]. The information provided by this type of surveys is relevant as supporting evidence to determine the relative importance of impediments but do not provide any evidence on the actual impediments encountered during development.

2.6 | Quality appraisal of the included studies

Kitchenham et al. [16] recommended that the candidate studies are assessed according to a predetermined checklist, to ensure that studies are impartially assessed for quality. The quality criteria used in this review are shown in Table 4 and are influenced by the criteria used by Atkins et al. [18] and Dybå and Dingsøyr [35].

The production and validity of the checklist itself [18] is a potential issue when assessing quality, as there is no consensus on what makes a study good and suitable for inclusion in a qualitative review [30]. We quickly realized that many experience reports have a "sense of rightness and feeling of comfort" [36] and "sound, well-grounded, justifiable, strong, and convincing" [37] arguments, which makes them suitable for inclusion in a qualitative study. The goal of the qualitative analysis and synthesis is to understand the phenomena, not to describe them [30], which allows us to consider weaker but still valuable evidence. Hence, we have performed the quality assessment but not excluded any studies based on the quality assessment.

2.7 | Data extraction and analysis

From each included primary study, we extracted all data relevant to the research objective of the review. Specifically, we collected all paragraphs of text containing claims on negative effects of, or impediments for, test automation. The data extraction was performed using the qualitative data analysis software NVivo [38], into which the full text reports were imported. The text was then analyzed using "thematic coding" as described by Robson [39].

In general, thematic analysis is an iterative method to identify common themes in the data, by classifying behaviors, events, activities or other concepts that are considered to influence the studied phenomenon. The themes are used to group segments of data, and these groups can then be examined to find repetitions, similarities, differences, anomalies, or connections to the theory that are further analyzed to provide the results from the study.

The analysis starts by identifying a number of potential codes for *coding*, or labeling, snippets of evidence. This can be done by defining a set of start codes based on prior understanding, incrementally during the work, or as a combination of both [19]. This is then applied to the body of evidence by dividing the material into strata belonging to the various codes.

After applying the code-set to the body of evidence, a series of iterative refinement takes place, where the strata are combined into themes. This is applied iteratively, increasing the level of abstraction for each step. Themes thereby form associations between the different sources included, which paints a picture of the reported body of knowledge. In our case, the themes resulting from the analysis are listed in Tables 5 and 6 and discussed in detail in Section 3.3. Further, an overarching view of the themes and the way in which they interact is given in Figure 4.

For a more-detailed description of the analysis method, we also refer to Cruzes and Dybå [19], who provide a detailed overview of how to perform [illegible] for its use in software engineering research.

TABLE 5 Organizational impediments

	Publications
Behavioural Effects	
Fear of Redundancy from Automation	[IP12]
General Negativity	[IP13],[IP14]
Lack of Management Support	[IP15],[IP9],[IP2]
Negativity Towards Automation	[IP16],[IP17],[IP3],[IP5]
Process Deviations	[IP17],[IP18],[IP12],[IP19],[IP11]
Processes Considered Too Complex for Automation	[IP7]
Shifting Blame Between Development and Test	[IP17],[IP20],[IP13]
Testing is not viewed as a natural part of the work	[IP14]
Too High Expectations	[IP16],[IP10],[IP14]
Expectations on Better Testing	[IP10]
Expectations on High Cost Reduction	[IP10]
Expectations on Quick ROI	[IP10],[IP17]
Business and Planning	
Cost of Ownership and Operation	[IP14]
Automation too Expensive for Small Projects	[IP3],[IP7],[IP21],[IP4],[IP5]
Implementation Costs	[IP22],[IP23],[IP3],[IP24],[IP7],[IP14],[IP5]
Maintenance Costs	[IP22],[IP16],[IP10],[IP17],[IP7],[IP14],[IP12],[IP25],[IP5]
Training Costs	[IP3],[IP7],[IP5]
Staffing	
Dependency to Key People	[IP1],[IP3],[IP2]
Testing by Non-testers	[IP15]
Steering	
Inadequate Automation Strategy	[IP22],[IP10],[IP1],[IP26],[IP17],[IP23],[IP4],[IP27]
Inadequate Test Strategy	[IP10],[IP28],[IP15],[IP18],[IP25],[IP19]
Lack of KPI for automation development	[IP17],[IP20],[IP7],[IP21],[IP25],[IP5],[IP19],[IP2]
Communication	[IP23]
Change Management of SUT Changes	[IP22],[IP16],[IP10],[IP6],[IP1],[IP26],[IP28]
Communication with Tool Suppliers	[IP2]
Roles and Responsibilities	[IP1],[IP26],[IP29],[IP15],[IP19],[IP11],[IP2]
Terminology	[IP4]
Lack of Time, People, and Funding	
Lack of Time for Automation	[IP22],[IP16],[IP1],[IP26],[IP17],[IP28],[IP29],[IP30] [IP7],[IP14],[IP21],[IP27],[IP5],[IP11]
Lack of Time for Testing	[IP28],[IP23],[IP14],[IP18],[IP31],[IP5]
Lead time between Delivery and Testing	[IP26],[IP25],[IP19]
No People Available	[IP15],[IP3],[IP24],[IP7],[IP21],[IP4],[IP25],[IP5],[IP19],[IP2]
No Test Environment Available	[IP3],[IP9],[IP24],[IP32],[IP33],[IP5]
No Time for Training	[IP5],[IP2]
Under-budgeting	[IP16],[IP15]
Skills	[IP2],[IP8]
Application Domain Skills	[IP10],[IP3],[IP8]
Automation Skills	[IP22],[IP16],[IP21],[IP34],[IP25],[IP2],[IP8]
General Development Skills	[IP2],[IP8]
Programming Skills	[IP16],[IP26],[IP29],[IP23],[IP15],[IP21],[IP8]
Steep Learning Curve	[IP16],[IP21],[IP11]
Testing Skills	[IP23],[IP15],[IP4],[IP25],[IP19],[IP2]

TABLE 6 Test system and system-under-test impediments

	Publications
Test System	
Environment Configuration	
Inadequate Environment Configuration Management	[IP16],[IP10],[IP15],[IP20],[IP9],[IP14],[IP4],[IP32],[IP34],[IP33],[IP8]
Problems with Configuration and Setup	[IP1],[IP26],[IP29],[IP14],[IP21],[IP18],[IP32],[IP34],[IP8]
Inadequate Development Practices	
Accumulating Technical Debt	[IP22],[IP10],[IP1],[IP30]
Ad Hoc Development	[IP22],[IP16],[IP10],[IP29],[IP30],[IP5],[IP11]
Inadequate Test System Documentation	[IP22],[IP21],[IP33],[IP11]
Missing Specifications	[IP16],[IP28],[IP15],[IP3],[IP18],[IP5],[IP19]
No Reuse of Test Code	[IP10],[IP29],[IP30],[IP5]
Testware Architecture	[IP10],[IP17],[IP33]
Untested Test Environment	[IP10],[IP9],[IP7],[IP32]
Quality Issues	
False Positives and Negatives	[IP16],[IP24],[IP7],[IP21],[IP4]
Fragile Test Scripts	[IP22],[IP16],[IP1],[IP26],[IP20],[IP21],[IP25],[IP35],[IP11]
Test Script Bugs	[IP16],[IP21]]
Test System Bugs	[IP22],[IP16],[IP3],[IP24],[IP5],[IP8]
Test System Instability	[IP22],[IP16],[IP9],[IP24],[IP21],[IP12],[IP32]
Technical Limitations	
Insufficient Scalability	[[IP9],[IP21]
Integration of Several Tools is Necessary	[IP22],[IP16],[IP20],[IP7],[IP18],[IP4],[IP35]
Limitations in Externally Sourced Tools	[IP22],[IP16],[IP26],[IP20],[IP24],[IP7],[IP14],[IP21],[IP4],[IP34],[IP35],[IP11]
Limited Test Tool Functionality	[IP16],[IP20],[IP14],[IP18]
Manual Interventions Needed	[IP10],[IP7],[IP4],[IP32],[IP33]
Problems with Capture and Replay Tools	[IP16],[IP1],[IP26],[IP20],[IP21],[IP4]
Slow Test System	[IP17],[IP28],[IP30],[IP20],[IP7],[IP12],[IP33]
Usability	
Difficult Analysis of Test Results	[IP10],[IP21],[IP18],[IP4],[IP33]
Low Usability	[IP22],[IP16],[IP20],[IP24],[IP21],[IP33],[IP11],[IP8]
IT Environment	
IT accesses	[IP32],[IP8]
IT infrastructure	[IP22],[IP8]
System Under Test	
Platform Limitations	[IP16],[IP28],[IP36],[IP3],[IP18],[IP31],[IP5]
SUT Complexity	[IP29],[IP3],[IP34],[IP5]
SUT Considered too Small for Automation	[IP28],[IP3],[IP5]
SUT Interfaces External Systems	[IP3],[IP5],[IP19]
SUT Quality	[IP1],[IP26],[IP4],[IP31],[IP19]
SUT Speed	[IP22]
SUT Testability	[IP16],[IP10],[IP17],[IP28],[IP36],[IP3],[IP14],[IP18],[IP34],[IP5]

3 | RESULTS

Considering the objectives for this systematic literature review:

- Objective 1: To identify scientifically reported software development impediments related to automated software testing.
  To fulfill objective 1, we extracted relevant evidence from 39 publications that were identified through a systematic literature search following the recommendations of Kitchenham et al. [16] and the Cochrane Handbook of Systematic Reviews [29]. The data extraction and analysis was performed by a meta-etnographic thematic analysis to produce the results presented in Section 3.3. Through the analysis, we identified and classified a large number of impediments that are presented in Tables 5 and 6. The impediments are grouped hierarchically, and the studies [illegible] evidence are listed per impediment type.

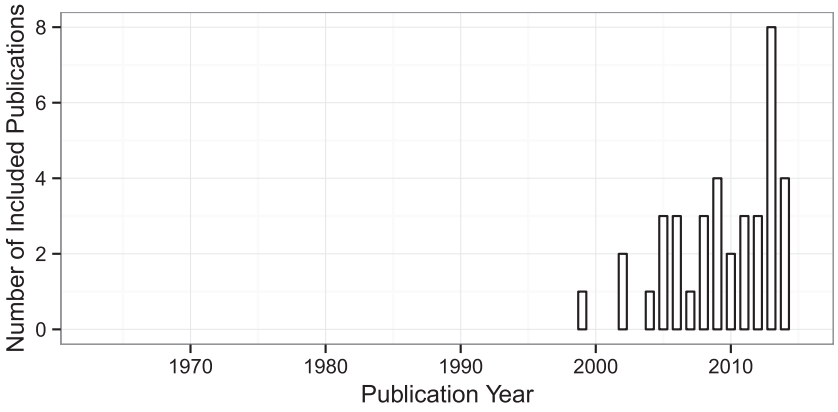

FIGURE 2 Included publications per year

- Objective 2: To explain the mechanisms leading to the impediments through a qualitative synthesis.
  To fulfill objective 2, we performed a qualitative synthesis that is presented in Section 3.3. The qualitative synthesis resulted in both the qualitative narrative, and a proposal of a socio-technical system model of interacting phenomena encountered during the use and implementation of test automation that is shown in Figure 4.

3.1 | Publication statistics

The unscreened combined initial result of the database searches, the journal runs, and the forward and backward searches consists of 13 060 publications, published over a time of 50 years: from 1964 to 2015-03-31.

As seen in the figure, most of the retrieved publications were published from the late 1980s and onwards. The publication rate is growing steadily, until the late 1990s, where there is an apparent slowdown during the "dot com" years, followed by a sharp increase from 2004 and onwards.

The increase after 2004 can likely be explained by an increased interest in empirical software engineering and testing. Kitchenham, Dybå, and Jørgensen published a paper promoting evidence-based software engineering [41] in 2004, the Empirical Software Engineering journal increased from four to six issues per year in 2007, and the International Conference on Software Testing, Verification and Validation (ICST) was first held in 2008. During the same period, the number of test certifications has increased dramatically. Statistics from ISTQB [42] report an increase of roughly 325 000 certifications between 2006 and 2014, compared to the previous eight year period from 1998 to 2006, when approximately 30 000 professionals received certifications from ISTQB.

The number of included publications are plotted in Figure 2, showing that all included publications in the synthesis were published between 1999 and 2014.

3.2 | Quality assessment

An overview of all included publications and the answer to the assessment questions is available in Table 7.

The absolute majority of the publications had at least a minimal description of the context that could be identified easily, but the contents and the quality of the content description varies greatly. We have not set a limit for the context quality, but noted that very few descriptions fulfill the requirements set by Petersen and Wohlin [43] or Dybå et al. [44]. This makes it harder to compare the study results, as some results may be valid only in specific contexts [28].

3.3 | Qualitative summary of the findings

This section presents our qualitative *interpretation* of the evidence collected through the systematic literature review.

Implementing test automation is difficult, and even if well prepared in advance, there are significant risks of failure [IP4]. We propose that this is because test automation is a "socio-technical challenge" [IP31] with a complex interaction between the various phenomena that emerge during use and implementation of the automation.

During the thematic analysis, it quickly became clear that there is an interdependence between the different themes that may form feedback loops that threaten to increase the magnitude of the impediment until the test automation initiative fails.

As an example, one of these feedback loops is shown isolated from its other dependencies in Figure 3. A change in any of the factors in the feedback loop will influence the other factors. Consider the case where automation is not viewed as useful. This will likely reduce the willingness to invest in the automation by allocating people and time to the work. Less time and people available to do the work will likely lead to a low quality automation. [illegible]ion is not as useful as high-quality automation, which reduces the perceived value and the feedback loop is closed.

On the basis of the systematic review and the experience of the researchers, we propose a system view of test automation as shown in Figure 4. During the thematic analysis of the primary studies, we identified interdependent relations between the themes that we propose may cause feedback in an automation project. To explain this, the figure was synthesised by identifying potential cause-effect relations in the primary studies, which

TABLE 7 Study quality assessment

			Quality criteria						
	Authors	Year	1	2	3	4	5	Type	Objective/Scope
[IP1]	Allott	1999	Y	Y	Y	N	Y	Experience Report	Experience report on three industrial test automation projects.
[IP2]	Rankin	2002	Y	Y	Y	N	Y	Experience Report	Experience report on the design and use of a test automation framework.
[IP3]	Fecko and Lott	2002	Y	Y	Y	N	Y	Experience Report	Experience report on test automation deployment and use in industry.
[IP4]	Persson and Yilmazturk	2004	Y	Y	Y	N	Y	Experience Report	Experience report on two industrial test automation projects.
[IP5]	Karlström et al.	2005	Y	Y	Y	Y	Y	Case Study	Empirical Evaluation of a test practice framework.
[IP6]	Damm et al.	2005	Y	Y	Y	N	Y	Experience Report	Introduce automated testing and TDD.
[IP7]	Berner et al.	2005	Y	Y	Y	N	Y	Experience Report	Experience report on automated testing.
[IP8]	Jeong	2006	Y	N	Y	N	Y	Experience Report	Experience report on automation of software build and deployment.
[IP9]	Esipchuk and Vavilov	2006	Y	N	Y	N	Y	Experience Report	Experience report on test automation for mobile phones.
[IP10]	Holmes and Kellogg	2006	Y	Y	Y	N	Y	Experience Report	Experience report on web browser test automation.
[IP11]	Martin et al.	2007	Y	Y	Y	Y	Y	Ethnography	Ethnographic study of software testing in a small business.
[IP12]	Rendell	2008	Y	Y	Y	N	Y	Experience Report	Experience report on test-driven development.
[IP13]	Garcia et al.	2008	Y	Y	Y	N	Y	Experience Report	Describe factors that impede verification and validation, based on author experience.
[[IP14]	Shaye	2008	Y	N	Y	N	Y	Experience Report	Experience report on changing to agile test methods.
[IP15]	Kasurinen et al.	2009	Y	Y	Y	Y	Y	Case Study	Explore issues related to software testing in practice.
[IP16]	Rooksby et al.	2009	Y	Y	Y	Y	Y	Ethnography	Ethnographic investigation of organizational effects on test practices.
[IP17]	Jakobsen et al.	2009	Y	Y	Y	N	Y	Experience Report	Report on introducing Scrum in a CMMI level 5 organization.
[IP18]	Karhu et al.	2009	Y	Y	Y	Y	Y	Case Study	Explore the factors that affect the use of automated testing.
[IP19]	Subramonian et al.	2009	Y	Y	Y	N	Y	Experience Report	Experience report on testing.
[IP20]	Pinheiro et al.	2010	Y	Y	Y	Y	Y	Action Research	Action research on testing of test environment configuration.
[IP21]	Kasurinen et al.	2010	Y	Y	Y	Y	Y	Qualitative Survey	Explore the state of test automation in industry and identify improvement opportunities.
[IP22]	Taipale et al.	2011	Y	Y	Y	Y	Y	Case Study	Explore the current state of test automation in industry.
[IP23]	Gruner and van Zyl	2011	Y	Y	Y	Y	Y	Case Study	Analysis of the test process of a small business.
[IP24]	Collins et al.	2012	Y	Y	Y	N	Y	Experience Report	Experience report on test automation.
[IP25]	Wiklund et al.	2012	Y	Y	Y	Y	Y	Case Study	Identify impediments for test automation.
[IP26]	Collins and de Lucena	2012	Y	Y	Y	N	Y	Experience Report	Experience report from an agile test team.
[IP27]	Collins et al.	2012	Y	Y	Y	N	Y	Experience Report	Experience report on automated testing.
[IP28]	Erfani Joorabchi et al.	2013	Y	Y	Y	Y	Y	Qualitative Survey	Investigate the main challenges for mobile software development.
[IP30]	Wiklund et al.	2013	Y	Y	Y	Y	Y	Case Study	Identify impediments in a newly formed agile software development organization.

(Continues)

TABLE 7 (Continued)

	Authors	Year	Quality criteria 1	2	3	4	5	Type	Objective/Scope
[IP31]	Toroi et al.	2013	Y	Y	Y	Y	Y	Case Study	Report on case studies conducted in three software businesses.
[IP32]	Alegroth et al.	2013	Y	Y	Y	Y	Y	Case Study	Explore industrial use of visual GUI testing.
[IP32]	Neely and Stolt	2013	Y	Y	N	N	Y	Experience Report	Experience report on introduction of continuous delivery.
[IP33]	Greiler et al.	2013	Y	Y	Y	Y	Y	Case Study	Implementation and evaluation of a static analysis tool for test fixtures.
[IP34]	Liebel et al.	2013	Y	Y	Y	Y	Y	Case Study	Investigate how GUI-based testing is performed in industry and identify related problems.
[IP35]	Ramler and Putschogl	2013	Y	Y	Y	N	Y	Experience Report	Experience report on building a test tool.
[IP36]	Wiklund et al.	2014	Y	Y	Y	Y	Y	Case Study	Identify problems encountered by test automation users.
[IP37]	Bauersfeld et al.	2014	Y	Y	Y	Y	Y	Case Study	Industrial evaluation of a test tool.
[IP38]	Kanij et al.	2014	Y	Y	Y	Y	Y	Survey	Survey on factors affecting software testers.
[IP39]	Mahmud et al.	2014	Y	Y	Y	Y	Y	Case Study	Describe the development of a testing tool and its deployment.

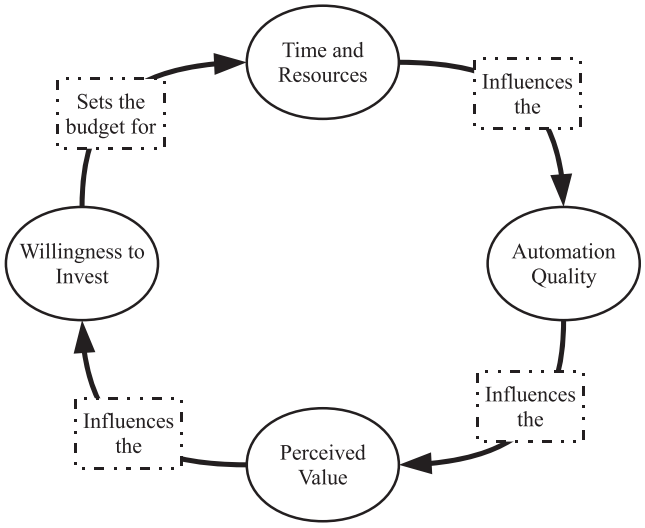

FIGURE 3 Feedback between connected phenomena

resulted in a very large graph. This graph was then further refined into the figure in the paper through combination of causes and effects into more and more refined themes until the presented result was reached.

Note that the figure (to be discussed in further detail in the following sections) does not contain all groups of impediments or phenomena, only those we consider to be most important for the socio-technical system. The square boxes represent the conceptual areas, and the arrows indicate influence from other areas. The rounded boxes represent the nature of the influence on the area they are located with.

3.3.1 | Behavioural effects

Our first finding is that people working in a test automation project have a significant influence on the outcome of the project [45] due to their behaviors [46], organizational culture [46], motivation [47], and expectations.

Process adherence

Test automation depends on stability and structure in the testing and processes to be successful [48]. It will be very difficult to successfully implement test automation if the testing in general is unstructured, considered optional [IP14], or there is a general negativity to processes and quality [IP13].

Tight deadlines and a high schedule pressure increases the risk for process deviations [IP17],[IP11] [49] that may cause problems later [IP18],[IP22],[IP30]. If the automated test is perceived as an impediment by its users, there is a risk that the testing is skipped [IP12] or that the [illegible] ned in favor of manual work [IP7].

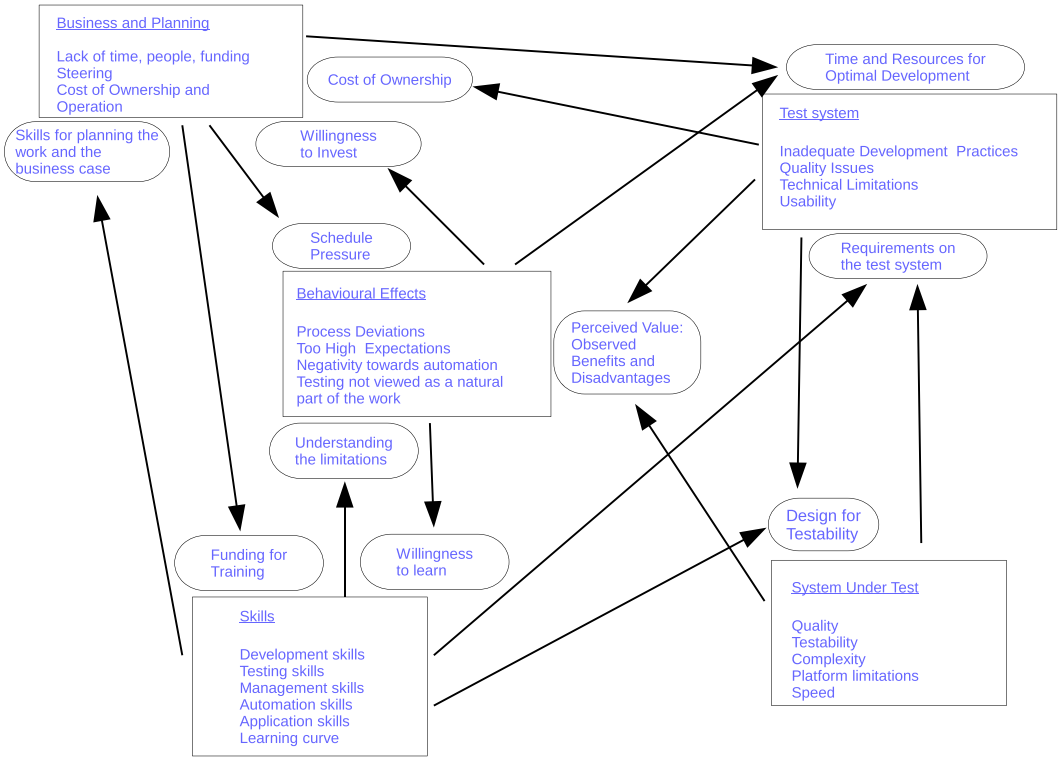

FIGURE 4 The socio-technical system of test automation

Organizational change

It is crucial to understand that the introduction of test automation at least partially is about implementing an organizational change. The introduction of automation has to be managed properly [IP16] to minimize negativity [IP16],[IP17],[IP3],[IP5] that may cause the automation project to fail. Lack of management support [IP15],[IP9],[IP2] will increase the risk for change-related problems.

When deploying automation, one must consider resistance to the change caused by automation [IP5], and the historically always present [50] fear of becoming unemployed due to automation [IP12]. Keeping employees motivated is an important factor in any change project and are not less important in an automation project [IP17][IP6].

Too high expectations

One of the major risks in a test automation project is that the expectations of the people involved are unrealistically high [IP14],[IP16],[IP10], which may result in abandonment of the automation if the expected results are not achieved within the expected time-frame [IP10]. Managing these expectations is very important for successful automation [IP16].

Automation is expensive to develop [IP22],[IP23],[IP3],[IP24],[IP7],[IP14],[IP5], and the likelihood of an immediate return on investment must be considered to be low [IP17],[51]. Focusing on rapid return on investment during deployment is likely to be detrimental to the maintainability of the system [IP10] and will cause costs in the long run.

Unrealistic expectations are largely caused by missing insight in what the automation can and cannot achieve [IP23],[IP4],[IP27]. Unrealistic expectations can be mitigated by change management [IP16], education, and investigations prior to the decision to automate. A well-developed automation strategy that can be used for deployment and development is a very important tool both to manage the automation project and to communicate what can and can not be done with the chosen tools [IP22],[IP10],[IP1],[IP26],[IP17],[IP23],[IP4] .

Unfulfilled expectations will likely have negative impact on the perceived value of the automation. If the test automation disappoint the stakeholders and the people involved in the work, it does not matter how much value it actually provide to the organization, and it will be considered as a wasteful activity.

Process Deviations

To invest time, money, and effort in something, both the business and its employees need to see the value of their tasks.

It is fairly well known that testing is among the first things in a development project that are ignored when the time is short and the deadlines are near [IP17],[IP18],[IP12],[IP19],[IP11],53]. The risk for this type of process deviations is increased if there is no focus on testing in the project [IP14],[IP15],[IP18]. If testing is neglected and not considered to be a valuable activity for the business, then by extension, the organization will not [illegible] automation either.

The high complexity of a test automation system is a risk to the perceived value of the automation. If manual testing is seen as easier to perform than writing the automated test scripts [IP5], the consequence may be that testers choose to work manually instead of automating tests, in particular if there is time pressure [IP22],[IP16],[IP1],[IP26],[IP17],[IP28],[IP29],[IP30],[IP7],[IP14],[IP21],[IP27],[IP5],[IP11].

3.3.2 | Business and planning

Many organizations exceed their software development budgets in general [53], and in light of this, it is not surprising that cost is considered to be a limiting factor for automation [IP15] [3] [IP14],[IP24],[IP7],[IP5],[IP22].

Cost of ownership and operation

Realizing that automation need to follow a reasonable business life cycle as opposed to getting the benefits immediately is an important precondition for successful automation [48,51]. Test automation projects are no different from other projects and need the same type of preparations [IP16]. Before automating anything, a business case [IP22] and an automation strategy need to be developed to assist decisions on what parts of the test scope shall be automated to what degree [IP27].

Automation too expensive for small projects

The properties of the product and features being developed are relevant to the possibility to get an acceptable return on the investment. There are examples of organizations that consider their project or products to be too small to automate efficiently [IP28],[IP3]. In small, short-lived, projects or products, it can and should be discussed if it is possible to get an acceptable ROI before the development ends [IP28],[IP3],[IP5],[IP7],[IP21],[IP4]. Setting up automation can be too costly compared to the rest of the development effort [IP28] [6]. In line with this, Kasurinen et al. [IP7] propose that the ratio between the resources needed for automation and the total availability of resources shall be used to guide the decisions on automation. Combined with work on divergent products [IP16] or on emerging platforms [IP28] [IP36], the organization may not see sufficient opportunities for long term reuse to motivate an investment in test automation [IP5] . For those situations, there could be other things than execution that can be automated with better long term results [IP10].

Lack of time, people, and funding

Damm et al. [IP17] note that projects in time-pressure tend to prioritize near-time goals and neglect activities that are important for the long-term success. This is reported for testing in general, where design organizations down-prioritize testing or remove test activities to meet delivery deadlines [IP11][53][IP27],[IP14],[IP15]. There are also reports of more or less permanent understaffing of the development teams with respect to testing [IP2],[IP19],[IP25]. This reflects the perceived importance of testing, as indicated by Liebel et al. [IP14] who reported that many developers in a studied organization considered test to be optional, something done if there was spare time at the end of the development cycle.

Considering automation, there are reports on choosing manual work instead of automation, if time is short [IP6] or if there are insufficient resources available for automation [IP7],[IP5]. This allows the testers to test immediately instead of establishing the infrastructure and test scripts. The result is likely accumulation of technical debt that will cause problems later [IP22]. Berner et al. summarized the approach in one of the projects they reported on as "we do not have the time to be efficient" [IP10].

3.3.3 | Skills

Test automation is often a complex activity that require specialist skills. It is likely that the replacement of manual work by automation will cause a major change in the testers daily work. Changing the work practices may require "major training" [IP3] if the introduction of automation is to be successful. Training requires both a budget and time and priority in the work schedule.

Diversity

A test automation project need implementation staff skilled in testing [IP4][3][IP19], developing software [IP10][52], using the system under test, and handling the automated tools. There is also a need for people handling planning-related tasks, such as product management for the test system, and general leadership and planning skills [IP4][IP10]. If these skills are not present in the project staff, the effects will be the same as on a badly staffed ordinary software development project: an increased risk for failure, slower work, lower quality, and budget overdrafts.

As an example, missing programming skills in the project may cause the organization to seek alternatives to programming. One such alternative is "capture and replay" [54] tools that offer a shorter implementation time for automated test scripts by recording manual tester actions for later execution. The resulting automated test scripts are usually more fragile [IP4] than test scripts that have been implemented by programming, which introduces a risk for more rework when the system under test changes [IP1],[IP26],[IP20]. Unless this has been anticipated and planned for, the rework may cause significant disappointment [IP21] in the automated testing, resulting in setbacks to the automation project [IP16].

Steep learning curve

Learning to automate tests is hard. Test automation tools may be too complex to use without significant training [IP3], and the learning curve may be very steep [IP21]. The likelihood for this is higher if the tools are intended for more advanced users than those actually using them [IP11]. Appropriate tool selection will influence the level of competence needed to work with the automation, but to do the tool selection, it is necessary to acquire relevant skills beforehand [3]. If the organization is unskilled, it might cause the introduction of technical debt that have to be refactored away later [IP22]. A high technical debt may make the return on investment slower than anticipated due to necessary rework to be able to perform necessary [illegible] the system under test behavior changes.

If manual testing is viewed as easier to perform than developing automated test scripts [IP5], the consequence may be that testers choose to work manually instead of automating tests, leading to a lower or completely missing return on investment from the automation.

Inadequate development practices

A critical insight is that test automation is software development [IP16] and need to be treated as such [IP10]. This means that there is a clear need for staff skilled in programming [IP16],[IP26],[IP29],[IP23],[IP15],[IP21] and an understanding of how to do software development [IP2],[IP8], to be able to do effective and efficient development of the test automation.

3.3.4 | Test system

During use, test tools are combined into a test bed, "an environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test" [55], that is used by the testers. Together with the automated test scripts, this forms a *test system*, which is used to test the application.

If the tools and the test systems cause problems, the automation provides little or no value to the testers and is perceived as nothing but a very significant impediment [IP2]. If the impediment is severe enough, it may lead to manual testing as a work-around [IP5] instead of solving the problems.

In the test tool domain, impediments spanning the entire tool life cycle, from sourcing tools to using tools, have been reported.

The primary subthemes in this category are development and maintenance of tools, tool selection, and configuration of the test beds and test system.

Testware architecture

A test automation system is typically at least as complex as the system under test [10,11] [IP7] [12], which makes it likely that the test automation software need to be designed, developed, and maintained in a mature [40] way, similarly to the software developed for delivery to a customer. Careless or ad hoc development of test automation software is likely to result in a lower quality test system [40]. Low quality in the test system may in turn, exactly as with any other software, result in problems for the users of the test system and a risk to the quality of the tested product. Defects in the test software may cause escaped defects [54] through false positives or false negatives [IP34],[IP9] in the testing.

Published literature suggests that test automation often is implemented without any consideration of architecture, standards, documentation or maintainability [IP10]. The shortcuts taken include not testing the test software [IP9], ignoring known problems [IP31], ignoring known technical debt [IP30], staffing with inexperienced temporary personnel without planning for competence transfer [IP5], and prioritizing rapid deployment over maintenance [IP7]. An automation project or activity need to be handled as "ordinary" software development [IP5],[IP11],[IP3],[IP10], and have adequate development standards, quality criteria, and a competent staff.

A contributing factor may be the lack of priority that commonly is exhibited for testing. There is evidence indicating that test tool development is considered to be a low priority activity, performed in whatever time there is to spare and without the careful actions involved in producing production software. Lack of time for test-related work due to other, higher prioritized activities, is relatively common and is discussed further in Section 3.3.2. Even if the lack of time is not the cause as such, the lower priority of test work indicate that it is seen as having less value than other work in the organization.

Untested test environment

If the test system quality is low, it will result in more expensive maintenance [IP7] and a lower quality of testing [IP31]. The consequences are likely a reduction in the overall perception of the value of the test automation as well as less time available for general improvement and feature additions to the automation. Considering situations such as the one reported by Karlström et al. [IP9], "in some situations it was hard to decide whether it was the test environment that caused a failure or the software under test," one can hardly blame those experiencing that kind of issues if they are wary of automation.

Limitations in externally sourced tools

Selecting the correct tools is very important. If the tool selection is done without concern for the process and the system under test, the result may be that the selected tools are unsuitable for the work [IP23] [IP11].

It is likely that a high level of customization in the application will make it harder to use automation efficiently [IP5]. If the application is customized [IP3], changing a lot [IP21] or running on an emerging platform [IP28], there will probably be issues with using generic tools, and more customization and in-house tool development has to be done. If the application is very customized, or operating on an emerging platform [IP28], there is an increased risk that tools and emulators used for testing are missing important features [IP18].

Reuse may be a way to mitigate the costs of automation, but one must consider that reuse have the potential to cause other problems in itself. Reuse is likely harder if the system under test is specialized [IP3], changing a lot [IP21] or running on an emerging platform [IP28]. These limitations for reuse are similar to the limitations for using prefabricated tools, which can be considered to be a special case of reuse. Prefabricated tools are discussed in Section 3.3.4. In addition to the purely functional considerations, many organizations develop tools ad hoc, as further discussed in Section 3.3.4, and this is a significant barrier for reuse [56].

An externally sourced tool is a tool not developed by its users. It may be a commercial tool, an open source tool, or simply a tool that is produced by another team in the company, sufficiently far away from the users of the tool to be perceived as a third party. The potential benefits of using externally sourced tools are easily imagined. These benefits include increased work focus for the product developers, who do not have to develop [illegible] sold product, and cost sharing by developing the tool once instead of everyone developing their own environment.

Using an externally sourced tool is fundamentally different from using a tool developed specifically for the system it is intended to test and introduces challenges that may cause significant problems for the users. The generic nature of a tool shared by many organizations may make it inadequate for the intended purpose [6] unless integrated with other tools [IP35],[IP11] or modified [IP7] to cover the gap. Commercial and open source tools have limited functionality [57] and may require modification to be useful. In the case of commercial tools, the modifications are likely impossible to do within the own organization and will require additional work by the supplier, which will cause more costs and increased lead times. It is likely that there will be communication issues [IP2] unless there is a close cooperation between the supplier and the users. Any problems with the communication between the supplier and the users will decrease both the actual value of the tools and the perceived value of the tools. This may have serious impact on the organizational efficiency. In our experience, people will start producing their own solutions if "being a customer" is seen as a larger impediment than developing a tool on your own.

Environment configuration

Configuration issues can be divided into 2 groups, problems relating to difficulties with performing the configuration and problems related to managing the configuration to be able to use known and relevant configurations for testing that are repeatable if necessary.

Ad hoc configuration of test tools and test beds introduces a test risk by potentially causing the tests to be executed on unknown configurations. This may lead to instable results with corresponding false positives/negatives [IP34],[IP9], and problems with repeatability of tests [IP15]. Since repeatability and confidence in the tests are among the key benefits of automation [6][48], the importance of reliable configuration should not be understated.

Configuration issues are likely to occur if there is a suboptimal organization [IP9] of the environment or if it is difficult to use [4] [IP21], [IP11],[IP14],[IP8]. Expectations on configurability are important: If the test system users expect the system to work without their involvement [IP17],[IP9],[IP11], it is likely that configuration and setup of the test system will be more difficult and take longer time than if it was considered a normal part of their work. Unexpected work is less likely to be considered during training and staffing and will form a double impediment, as the necessary skills and people have to be secured before the work can be done.

Resolving configuration and setup issues may consume a large part of the time and resources dedicated to testing, to a point where the automation can be considered to be useless [IP29],[IP18],[IP32].

3.3.5 | System under test

The properties of the system under test are very important for successful automated testing and will influence the design and operation of the test system.

SUT speed

One of the primary expectations on automation compared to manual work is increased speed. However, automated test execution speed cannot be faster than the reaction time of the system under test. This means that if the system under test is slow, the automated test execution will also be slow [IP22], and the perceived value of the automation risk becoming lower due to high expectations on speed.

SUT testability

Low testability in the system under test, with respect to automated testing, may cause problems [IP17] and potentially expensive workarounds to be able to implement the automated testing. This is particularly a problem for interfaces not designed for machine-machine interactions [IP28], such as graphical user interfaces [IP16],[IP34],[IP5]. In the case of testing a graphical user interface, it may be more cost-effective to improve the testability of the system under test and introduce a testing interface for functionality, than to implement and maintain functional testing through the graphical user interface [IP10].

Platform limitations

Targeting the same functionality towards different operating systems [IP5],[IP3], languages [IP33] or platforms [IP28] is likely to introduce impediments for the automated testing. Variations in look and feel, interfaces, and testability may cause problems when interfacing the system under test from an automated test execution system [IP28]. A prime example is development of mobile applications for smartphones, tablets, and similar. Mobile applications is a rapidly expanding field [58], and the resulting products are deployed in a highly fragmented universe of different devices from different manufacturers [IP28]. Similar phenomena can be encountered in development of desktop software as well, when there are differences in the exposed interfaces, or if one expects to use the same test automation system for fundamentally different products without adaptations [IP5],[IP3].

4 | THREATS TO VALIDITY

4.1 | Construct validity

Recall is the percentage of all relevant document in the searched databases that have been retrieved, and the precision is the percentage of retrieved documents that are relevant [20]. Having low recall corresponds to a failure to retrieve all or most relevant papers.

Searches with low precision require removal of many candidates, which increases the risk for misinterpretation or researcher fatigue. Given that approximately 99.7% of the initial search yield through manual screening, as shown by Figure 2, there is a risk that we have failed

to identify primary studies with poor title or abstract during our screening process. Inadequate keywords and terminology is a well-known issue when performing systematic literature reviews [28], and we know that relevant publications were missing completely from the initial corpus. We had previous knowledge of one relevant study by Karlström et al. [IP9] that was not found by the search strings, as reported in Section 2.4. This validity issue was mitigated by hand-searching a selection of relevant venues as described in Section 2.3.

4.2 | Internal validity

In identifying impediments, there is always a risk that the primary studies have not reported the actual impediments, but instead the impediments that were perceived at the time of the study. In that case, this has carried over to our synthesis. A further threat to internal validity is the substantial industrial experience of the authors. While this industrial experience has served to facilitate analysis and conceptualization, it might also unconsciously have impacted data extraction and analysis, e.g., judgment of relevance of certain potentially domain-specific impediments.

4.3 | External validity

One readily identifiable challenge with performing a systematic literature review in this field of research is the lack of evidence. In total, 39 relevant publications were included after screening, of which 18 are experience reports. This is on par with the 25 publications analyzed by Rafi et al. [6] in their review of benefits and limitations of automated software testing.

Several publications are partially overlapping, which makes the actual body of evidence smaller than it appears. Publications [IP3],[IP24],[IP7] and [IP5] draw from the same source material. The publications are written from different points of view and with different analyses of the material, but the contexts are the same and the presented evidence is very similar. Also, publications [IP1] and [IP26] at least partially share the same context.

Publication bias occurs when the published research is unrepresentative for the research actually performed in the area [59]. This introduces a risk of drawing the wrong conclusions when performing secondary research. A small body of evidence, such as the one used for this review, is more sensitive to bias problems. Kitchenham et al. [16] proposes that there is an increased risk for publication bias in industrial research that is viewed as potentially sensitive. Test research is arguably sensitive, as the testing is related to the product quality, and as such, is directly related to the bottom line of any software business.

However, much research is anonymous, and there is a reasonable spread of the severity of the issues reported in the included publications. The major risk of bias is likely in evidence drawn from experience reports, as many experience reports paint a very positive view of the situation. All considered, we consider it likely that there is a sufficient mix of both trivial and significant impediments in the published literature.

4.4 | Reliability

The limited volume of evidence means that the basis for the synthesis can be questioned. Many of the publications would not have been included if one were to follow strict inclusion criteria and only accepted research reports. In our opinion, we have captured a better picture of the practice by also including the experience reports in the analysis.

The search approach and screening influences the reliability as well, due to the large part of manual selection involved. Noblit and Hare [17] point out that "[a] synthesis reveals as much about the perspective of the synthesizer as it does about the substance of the synthesis." What this means is that the synthesized result is highly dependent on the researchers involved, and is only one of the possible interpretations of the data [30]. This is a threat to the validity of the study, as it is likely that researchers with different background or perspective will reach different conclusions.

5 | DISCUSSION AND FUTURE WORK

In this section, we discuss the implications of the review results for research and practitioners and propose future work based on the results and meta-observations during the review.

5.1 | Implications for researchers

A related literature review on automated software testing was undertaken by Rafi et al. [6]. While some of the impediments of automated software testing identified pointed out in our work were also identified as limitations in that review, our work extends the work of Rafi et al. in the following ways: First, instead of only listing limitations or impediments that may arise in automated software testing, we elaborate on how these impediments manifest and why they do so. Second, we describe an explanatory socio-technical system model of how the impediments affect each other in Figure 4. Third, while the review by Rafi et al. was presented in a very summarized form (the search process, analysis and results were presented in less than two pages), we describe the our review and analysis process in depth.

For researchers, the categorization, the publication overview, and the synthesized results are useful as a basis for future research and to understand the state-of-the-art and the state-of-the-practice as of the time of the review.

Terminology is a known issue in software engineering research [4,23–28] and a threat to the reliability of empirical research. Our categorization, presented in Tables 5 and 6, provide a framework for classification of test automation impediments that can be used during future research in

The publication overview reveals that there is a shortage of empirical evidence in the research area. There are few *research* publications, 18 of 39 of the analyzed publications are experience reports. This indicates that there is both a need for, and a clear opportunity to perform, industry-relevant research in the area: There is a high practitioner interest and a lack of research publications.

Our interpretation of the collected evidence, based on our experiences from industry and research, is presented in the synthesis and the model in Figure 4. The synthesized narrative together with the system can be used both to quickly gain an understanding for the area and to direct future research.

5.2 | Implications for practitioners

It is clear from the review that there are similarities in experienced impediments between different businesses, product types, and development models. Hence, we recommend anyone that use, or plan to use, test automation, to consider the findings in this paper. Our results are useful to gain a deeper insight into the complexity and challenges of test automation, to be able to implement and use test automation in a more effective and efficient way.

Understanding a problem is the first part of a solution for the problem. In particular, we propose that the socio-technical system model of test automation presented in Figure 4 will be useful for development organizations that implement test automation, where the model can be used to understand and manage the automation by monitoring and controlling the interacting phenomena described by the model.

5.3 | Meta-observations on the review process

A meta-conclusion from the review is that conducting systematic literature reviews in software engineering are difficult. Many challenges were encountered during the review process: unclear terminology [28], influential databases did not provide machine readable search results, and a large number of irrelevant documents were returned by the searches which required extensive screening. These impediments need to be removed to enable more high-quality reviews in a shorter time in the future.

Barroso et al. [20] state that the main threat to the validity of a systematic literature review is the failure to conduct a sufficiently exhaustive search. This is mitigated by relaxing the search criteria to include more potentially relevant publications in the initial corpus and perform a screening to identify what actually is relevant to the study [29]. A consequence of this is that there is a need for extensive, manual, postprocessing of the searches, which also may introduce bias to the study. An alternate approach to searching and screening is "snowballing," searching through the graph of references starting with a set of key publications. This approach was investigated by Jalali and Wohlin [25] who concluded that it could be a more efficient approach, in particular when the search query contain general terms that risk matching many irrelevant publications.

Given the amount of work needed to perform this review and the resulting yield, less than 0.3% of the originally retrieved publications, we see an urgent need to improve how publications are classified and to improve the search precision in the publication databases.

5.4 | Future work

In general, we see a need for empirical research in industry to increase the body of evidence related to *actual use* of test automation. This observation is based on the relatively small number of relevant research publications identified during this study. In our experience, this will require close cooperation between industry and academia, as the type of empirical research needed to provide deeper insights into the actual problems experienced by software developers need to be conducted close to the people performing the work [61].

A large part of the impediments are of an organizational, and not technical, nature, as shown in Table 5. Many of these impediments can be suspected to be caused by an ad hoc approach to the management and handling of the test automation, even in organizations that are mature in their handling of their commercial products. This class of impediments are reported from a wide diversity of contexts and appears to be sufficiently general in nature to warrant further investigation. What are the reasons for the apparent lack of focus on testing and test automation, and how can the situation be improved?

There is a lack of test process improvement and assessment models suitable for test automation [IP11]. To address this, we propose the development of an improvement model in the form of a measurement system similar to a balanced scorecard [60] that is based on the model proposed in Figure 4. The implementation of the measurement system is proposed to be a combination of objective measurement, as proposed by Eldh et al. [12], and self-assessment. The validation of the measurement system will require further industrial empirical research, which in addition to validating the model, also will contribute to the general body of evidence in the test automation area.

6 | CONCLUSIONS

In this paper, we report on a systematic literature review, searching for publications providing information on impediments related to automated software testing. The retrieved publications were analyzed to propose a categorization of impediments and to perform a qualitative synthesis of the

The qualitative synthesis is based on the evidence collected through the review and presents our interpretation of the state of the practice as described by the publications included in the review. The synthesized results as presented by the narrative in Section 3.3, and the model in Figure 4, builds on the collected evidence and our experience from industry and research.

This study proposes an explanatory model for the impediments and their interaction in Figure 4. The explanatory model relates the impediments to each other, and to the business of software development and testing in the large. The explanatory model highlights areas of interest for future research and enables a deeper understanding of how to manage impediments in an automation project.

It is important to make software engineering research useful and available to practitioners [2]. One way to make research more available to practitioners is to create assessment and improvement models that can be used to guide the daily work in a software development business. We propose that there is a need for an assessment and improvement model specifically aimed at software test automation. Hence, we propose further research to turn the explanatory model in Figure 4 into an assessment and improvement model for software development organizations working with test automation.

ACKNOWLEDGEMENTS

The work was performed within the frame of ITS-EASY, an industrial research school in Embedded Software and Systems, affiliated with the School of Innovation, Design and Engineering at Mälardalen University, Sweden. The work was funded by Ericsson AB, Mälardalen University, and the Knowledge Foundation.

ORCID


Kristian Wiklund http://orcid.org/0000-0001-8228-813X

INCLUDED PRIMARY STUDIES

[IP1] E. Collins, A. Dias-Neto, and VFD Jr. Lucena, *Strategies for Agile Software Testing Automation: An Industrial Experience*, 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, Izmir, Turkey, 2012, pp. 440–445.

[IP2] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundqvist, *Impediments in Agile Software Development: An Empirical Investigation*, Product Focused Software Process Improvement (PROFES), Springer Verlag, Paphos, Cyprus, 2013, pp. 35–49.

[IP3] K. Karhu, T. Repo, O. Taipale, and K. Smolander, *Empirical Observations on Software Testing Automation*, 2009 International Conference on Software testing Verification and Validation, IEEE, Denver, Colorado, USA, 2009, pp. 201–209.

[IP4] C. Persson and N. Yilmazturk, *Establishment of Automated Regression Testing at ABB: Industrial Experience Report on 'Avoiding the Pitfalls'*, Proceedings. 19th International Conference on Automated Software Engineering, 2004, IEEE, Linz, Austria, 2004, pp. 112–121.

[IP5] O. Taipale, J. Kasurinen, K. Karhu, and K. Smolander, *Trade-off between automated and manual software testing*, Int. J. Syst. Assur. Eng. Manage. **2** (2011), no. 2, 1–12.

[IP6] E. Collins, G. Macedo, N. Maia, and A. Dias-Neto, *An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment*, 2012 IEEE Seventh Int. Conf. Global Software Eng. (2012), 190–194.

[IP7] J. Kasurinen, O. Taipale, and K. Smolander, *Software test automation in practice: empirical observations*, Adv. Software Eng. **2010** (2010), 1–13.

[IP8] K. Wiklund, D. Sundmark, S. Eldh, and K. Lundqvist, *Impediments for Automated Testing - An Empirical Analysis of a User Support Discussion Board*, Software Testing, Verification and Validation (ICST), IEEE International Conference on, Cleveland, Ohio, USA, 2014, pp. 113–122.

[IP9] D. Karlström, P. Runeson, and S. Nordén, *A minimal test practice framework for emerging software organizations*, Software Test Verification Reliab. **15** (2005), no. 3, 145–166.

[IP10] S. Berner, R. Weber, and R. Keller, *Observations and lessons learned from automated testing*, Proceedings of the 27th International Conference on Software Engineering, ACM;Seattle, Washington, USA, 2005, pp. 571–579.

[IP11] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, *Technical Debt in Test Automation*, Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on, IEEE, Montréal, Canada, 2012, pp. 887–892.

[IP12] S. Neely and S. Stolt, *Continuous Delivery? Easy! Just Change Everything (Well, Maybe It Is Not That Easy)*, 2013 Agile Conf. (2013), 121–128.

[IP13] Y-K Jeong, *Project Automation: Case study on the NeOSS Project*, 5th IEEE/ACIS Int. Conf. Comput. Inf. Sci. 1st IEEE/ACIS Int. Workshop Compon.-Based Software Eng. Software Archit. Reuse (ICIS-COMSAR'06) **i** (2006), 259–264.

[IP14] G. Liebel, E. Alegroth, and R. Feldt, *State-of-Practice in GUI-based System and Acceptance Testing: An Industrial Multiple-Case Study*, 2013 39th Euromicro Conf. Software Eng. Adv. Appl. (2013), 17–24.

[IP15] S. Gruner and J. van Zyl, *Software Testing in Small IT Companies : A (not only) South African Problem*, South Afr. Comput. J. **47** (2011), 7–32.

[IP16] S. K. Allott, in *Automate Your Tests-You Won't Regress It!*. Seventeenth Annual Pacific Northwest Software Quality Conference, Portland, Oregon, USA, 1999, pp. 135–157.

[IP17] L. Damm, L. Lundberg, and D. Olsson, *Introducing test automation and test-driven development: An experience report*, Electron. Notes Theor. Comput. Sci. **116** (2005), 3–15.

[IP18] D. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville, *'Good' Organisational Reasons for 'Bad' Software Testing: An Ethnographic Study of Testing in a Small Software Company*, 29th Int. Conf. Software Eng. (ICSE'07) (2007), 602–611.

[IP19] T. Toroi, A. Raninen, and L. Vaatainen, *Identifying Process Improvement Targets in Test Processes: A Case Study*, 2013 IEEE Int. Conf. Software Maintenance

[IP20] A. Holmes and M. Kellogg, *Automating Functional Tests Using Selenium*, Agile 2006 (Agile'06) (2006), 270–275.

[IP21] J. Mahmud, A. Cypher, E. Haber, and T. Lau, *Design and industrial evaluation of a tool supporting semi-automated website testing*, Software Test Verification Reliab. **24** (2014), no. 1, 61–82.

[IP22] E. Alegroth, R. Feldt, and H. H. Olsson, *Transitioning Manual System Test Suites to Automated Testing: An Industrial Case Study*, 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, IEEE; Luxembourg, Luxembourg, 2013, pp. 56–65.

[IP23] J. Garcia, A. de Amescua, M. Velasco, and A. Sanz, *Ten Factors that Impede Improvement of Verification and Validation Processes in Software Intensive Organizations*, Software Process. Improv. Pract. **13** (2008), 335–343.

[IP24] J. Kasurinen, O. Taipale, and K. Smolander, *Analysis of Problems in Testing Practices*, 2009 16th Asia-Pacific Software Engineering Conference, IEEE, Los Alamitos, CA, USA, 2009, pp. 309–315.

[IP25] S. D. Shaye, Transitioning a Team to Agile Test Methods, Agile Conference, 2008, G Melnick, P Kruchten, and M Poppendieck, (eds.), IEEE; Toronto, Ontario, Canada, 2008, pp. 470–477.

[IP26] E. F. Collins and V. F. de Lucena, *Software Test Automation practices in agile development environment: An industry experience report*, 2012 7th Int. Workshop Autom. Software Test (AST) (2012), 57–63.

[IP27] A. Rendell, *Effective and Pragmatic Test Driven Development*, Agile 2008 Conf. (2008), 298–303.

[IP28] M. Erfani Joorabchi, A. Mesbah, and P. Kruchten, *Real Challenges in Mobile App Development*, ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Baltimore, MD, USA, 2013, pp. 15–24.

[IP29] M. A. Fecko and C. M. Lott, *Lessons learned from automating tests for an operations support system*, Software: Pract. Experience **32** (2002), no. 15, 1485–1506.

[IP30] M. Greiler, A. van Deursen, and M. A. Storey, *Automated Detection of Test Fixture Strategies and Smells*, *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, IEEE; Luxembourg, Luxembourg, 2013, pp. 322–331.

[IP31] J. Rooksby, M. Rouncefield, and I. Sommerville, *Testing in the Wild: The Social and Organisational Dimensions of Real World Practice*, Comput. Supported Cooperative Work (CSCW) **18** (2009), no. 5-6, 559–580.

[IP32] C. Pinheiro, V. Garousi, F. Maurer, and J. Sillito, *Introducing Automated Environment Configuration Testing in an Industrial Setting*, 22nd International Conference on Software Engineering and Knowledge Engineering, SEKE, Redwood City, CA, USA, 2010, pp. 186–191.

[IP33] C. Rankin, *The Software Testing Automation Framework*, IBM Syst. J. **41** (2002), no. 1, 126–139.

[IP34] R. Ramler and W. Putschogl, *A Retrospection on Building a Custom Tool for Automated System Testing*, 2013 IEEE 37th Annu. Comput. Software Appl. Conf. (2013), 820–821.

[IP35] V. Subramonian, E. Cheung, and G. Karam, *Automated testing of a converged conferencing application*, 2009 ICSE Workshop on Automation of Software Test, IEEE; Vancouver, British Columbia, Canada, 2009, pp. 101–105.

[IP36] I. Esipchuk and D. Vavilov, *PTF-based Test Automation for JAVA Applications on Mobile Phones*, 2006 IEEE Int. Symp. Consum. Electron. (2006), 1–3.

[IP37] C. R. Jakobsen and J. Sutherland, *Scrum and CMMI Going from Good to Great*, 2009 Agile Conf. (2009), 333–337.

[IP38] S. Bauersfeld, T. E. J. Vos, N. Condori-Fernandez, A. Bagnato, and E. Brosse, *Evaluating the TEST AR tool in an industrial case study*, Proc 8th ACM/IEEE Int. Symp. Empirical Software Eng. Meas - ESEM '14 (2014), 1–9.

[IP39] T. Kanij, R. Merkel, and J. Grundy, *A Preliminary Survey of Factors Affecting Software Testers.* 2014 23rd Australian Software Engineering Conference, Sydney, Australia, 2014, pp. 180–189.

REFERENCES

1. *TPI Next®: Business Driven Test Process Improvement*, UTN Publishers, 2009.
2. B. W. Boehm, *Software Engineering*, IEEE Trans. Comput. **C-25** (1976), no. 12, 1226–1241.
3. S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen, *A preliminary survey on software testing practices in Australia,* 2004 Australian Software Engineering Conference. Proceedings, Melbourne, Australia, 2004, pp. 116–125.
4. V. Garousi and J. Zhi, *A survey of software testing practices in Canada*, J. Syst. Software. **86** (2013), no. 5, 1354–1376.
5. N. Kerzazi and F. Khomh, *Factors impacting rapid releases.* Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14, ACM Press, New York, New York, USA, 2014, pp. 1–8.
6. D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä, *Benefits and limitations of automated software testing: Systematic literature review and practitioner survey*. Automation of Software Test, 7th International Workshop on, Zurich, Switzerland, 2012, pp. 36–42.
7. S. Stolberg, *Enabling Agile testing through continuous integration.* Agile Conference, 2009, Y. Dubinsky, T. Dyba, S. Adolph, and A. Sidky, (eds.), IEEE; Chicago, IL, USA, 2009, pp. 369–374.
8. B. Haugset and G. K. Hanssen, *Automated acceptance testing: A literature review and an industrial case study*, Agile 2008 Conf. (2008), 27–38.
9. S. Chopra, *Implementing Agile in old technology projects.* Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization, Noida, India, 2014, pp. 1–4.
10. T. Bhat and N. Nagappan, *Evaluating the efficacy of test-driven development.* International Symposium on Empirical Software Engineering - ISESE '06, ACM Press, New York, New York, USA, 2006, pp. 356.
11. B. V. Rompaey and S. Demeyer, *Estimation of test code changes using historical release data*, 2008 15th Working Conference on Reverse Engineering, Antwerp, Belgium, 2008, pp. 269–278.
12. S. Eldh, K. Andersson, A. Ermedahl, and K. Wiklund, *Towards a test automation improvement model (TAIM).* TAIC-PART 2014, IEEE, Cleveland, OH, 2014, pp. 337–342.
13. N. Brown, Y. Cai, and Y. Guo, et al., *Managing technical debt in software-reliant systems.* Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, ACM, New York, New York, USA, 2010, pp. 47–52.

14. Scrum Alliance - Glossary of Scrum Terms, 2012. http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms.
15. S. Anand, E. Burke, and T. Y. Chen, et al., *An orchestrated survey on automated software test case generation*, J. Syst. Software. **86** (August 2013), no. 8, 1978–2001.
16. B. Kitchenham, S. Charters, and D. Budgen, et al., *Guidelines for performing systematic literature reviews in software engineering*, Technical Report EBSE 2007-001,EBSE Technical Report EBSE-2007-01, Dept of Computer Science, University of Durham, Durham, UK, 2007.
17. G. W. Noblit and D. Hare, *Meta-Ethnography: Synthesizing qualitative studies*, SAGE Publications, Inc, 1988.
18. S. Atkins, S. Lewin, H. Smith, M. Engel, A. Fretheim, and J. Volmink, *Conducting a meta-ethnography of qualitative literature: Lessons learnt*, BMC Med. Res. Method. **8** (2008), 21.
19. D. S. Cruzes and T. Dyba, *Recommended steps for thematic synthesis in software engineering*. 2011 International Symposium on Empirical Software Engineering and Measurement. IEEE; Banff, AB, Canada, 2011, pp. 275–284.
20. J. Barroso, C. J. Gollop, M. Sandelowski, J. Meynell, P. F. Pearce, and L. J. Collins, *The challenges of searching for and retrieving qualitative studies*, Western J. Nursing Res. **25** (2003), no. 2, 153–178.
21. H. M. Cooper, *Synthesizing Research: A Guide for Literature Reviews*, 3rd ed., Sage Publications, Inc., 1998.
22. H. Zhang, M. A. Babar, and P. Tell, *Identifying relevant studies in software engineering*, Inf. Software Technol. **53** (2011), no. 6, 625–637.
23. J. Rader, E. J. Morris, and A. W. Brown, *An investigation into the state-of-the-practice of CASE tool integration*, Software Engineering Environments Conference, 1993. Proceedings, Reading, UK, 1993, pp. 209–221.
24. M. Grindal, J. Offutt, and J. Mellin, *On the testing maturity of software producing organizations: Detailed data. Technical Report*. Department of Information and Software Engineering, George Mason University. Fairfax, VA, 2006. http://www.ise.gmu.edu/techrep.
25. S. Jalali and C. Wohlin, *Systematic literature studies: Database searches vs. backward snowballing*. Proceedings of the 2012 6th ACM_IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Lund, Sweden, 2012, pp. 29–38.
26. R. Harrison and M. Wells, *A meta-analysis of multidisciplinary research*. Empirical Assessment in Software Engineering (EASE), Keele, Staffordshire, UK, 2000. Appendix 1.
27. F. García, et al., *Towards a consistent terminology for software measurement*, Inf. Software Technol. **48** (2006), no. 8, 631–644.
28. C. Wohlin, *Writing for synthesis of evidence in empirical software engineering*, Proc 8th ACM/IEEE Int. Symp. Empirical Software Eng. Meas - ESEM '14. (2014), 1–4.
29. J. P. T. Higgins and S. Green, *Cochrane Handbook for Systematic Reviews of Interventions Version 5.1.0 [updated March 2011]*. The Cochrane Collaboration, 2011, available at https://www.handbook.cochrane.org
30. F. Toye, K. Seers, N. Allcock, M. Briggs, E. Carr, and K. Barker, *Meta-ethnography 25 years on: Challenges and insights for synthesising a large number of qualitative studies.*, BMC Med. Res. Method. **14** (2014), no. 1, 80.
31. J. Cohen, *A coefficient of agreement for nominal scales*, Educational Psychological Meas. **20** (1960), no. 1, 37–46.
32. R. Campbell, et al., *Evaluating meta-ethnography: A synthesis of qualitative research on lay experiences of diabetes and diabetes care*, Social Sci. Med. **56** (2003), no. 4, 671–684.
33. F. Scheuren, *What is a Survey*, 2004. https://www.whatisasurvey.info/download.htm
34. N. Schwarz, *Self-reports: How the questions shape the answers*, Am. Psychologist. **54** (1999) 93–105.
35. T. Dyba and T. Dingsoyr, *Empirical studies of agile software development: A systematic review*, Inf. Software Technol. **50** (2008), no. 9-10, 833–859.
36. M. Sandelowski and J. Barroso, *Reading qualitative studies*, Int. J. Qualitative Methods. **1** (2002), no. 1, 74–108.
37. S. Kvale, *The social construction of validity*, Qualitative Inquiry. **1** (1995), no. 1, 19–40.
38. *NVivo Qualitative Data Analysis Software, Version 10*, QSR International Pty Ltd., 2012.
39. C. Robson, *Real World Research*, 3rd ed., John Wiley & Sons, 2011.
40. M. C. Paulk, B. Curtis, M. B. Chrissis, and C. V. Weber, *Capability maturity model for software, version 1 . 1*, Carnegie Mellon Univ. (1993), 82.
41. B. A. Kitchenham, T. Dybå, and M. Jørgensen, *Evidence-based software engineering*. 26th International Conference on Software Engineering (ICSE'04), Edinburgh, UK, 2004, pp. 273–281.
42. *ISTQB Facts and Figures*. http://www.istqb.org/about-istqb/facts-and-figures.html
43. K. Petersen and C. Wohlin, *Context in industrial software engineering research*. International Symposium on Empirical Software Engineering and Measurement, Orlando, Florida, USA, 2009, pp. 401–404.
44. T. Dybå, D. I. Sjøberg, and D. S. Cruzes, *What works for whom, where, when, and why?*. Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '12, Lund, Sweden, 2012, pp. 19.
45. P. Lenberg, R. Feldt, and L. G. Wallgren, *Towards a behavioral software engineering*. Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE 2014, Hyderabad, India, 2014, pp. 48–55.
46. C. Briggs and P. Little, *Impacts of organizational culture and personality traits on decision-making in technical organizations*, Syst. Eng. **11** (2008), no. 1, 15–26.
47. S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, *Motivation in software engineering: A systematc literature review*, Eng. September (2007), 1–30.
48. M. Fewster and D. Graham, *Software Test Automation: Effective Use of Test Execution Tools*, Addison-Wesley, 1999.
49. Y. Kwak and J. Stoddard, *Project risk management: Lessons learned from software development environment*, Technovation. **24** (2004), 915–920.
50. K. Marx, *Das Kapital 1. Kritik der Politischen Ökonomie. Der Produktionsprozess des Kapitals*, 1867.
51. C. Kaner, *Pitfalls and strategies in automated testing*, Comput. **30** (1997), no. 4, 114–116.
52. Z. Shaheen, A. Rauf, and B. Hameed, *Automated software testing: An insight into local industry*, AWERProcedia Inf. Technol. Comput. Sci. **04** (2013), 408–415.
53. R. Torkar and S. Mankefors, *A survey on testing and reuse*. International Conference on Software–Science, Technology & Engineering, IEEE Comput. Soc; Herzlia, Israel, 2003, pp. 164–173.

[illegible] l, *Standard Glossary of Terms used in Software Testing International Software Testing Qualifications Board*, Version 2, ISTQB, 2014.

55. IEEE Standard Glossary of Software Engineering Terminology. Technical Report, 1990.
56. C. Jones, *Economics of software reuse*, Comput. **27** (1994), no. 7, 106–107.
57. M. Grechanik, Q. Xie, and C. Fu, *Creating GUI testing tools using accessibility technologies.* 2009 International Conference on Software Testing, Verification, and Validation Workshops, Denver, Colorado, USA, 2009, pp. 243–250.
58. S. Perez, *iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads To Date*, 2014. http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/
59. H. R. Rothstein, A. J. Sutton, and M. Borenstein, Publication bias in meta-analysis, *Publication Bias in Meta-Analysis: Prevention, Assessment and Adjustments*, Wiley, Vol. **4**, 2006, pp. 1–7.
60. R. S. Kaplan and D. P. Norton, *Using the balanced scorecard as a strategic management system*, Harvard Bus. Rev. **74** (1996), no. 1, 75–85.
61. K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, *Can we do useful industrial software engineering research in the shadow of Lean and Agile?* Conducting Empirical Studies in Industry, 1st Intl Workshop on, San Fransisco, CA, USA, 2013, pp. 67–68.

How to cite this article: Wiklund K, Eldh S, Sundmark D, Lundqvist K. Impediments for software test automation: A systematic literature review. *Softw Test Verif Reliab.* 2017;27:e1639. https://doi.org/10.1002/stvr.1639