Master's thesis

Master's Programme in Computer Science

Industrial Surveys on Software Testing Practices: A Literature Review

Kim Bäckström

January 2, 2022

Faculty of Science

University of Helsinki

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Kim Bäckström

Työn nimi — Arbetets titel — Title

Industrial Surveys on Software Testing Practices: A Literature Review

Ohjaajat — Handledare — Supervisors

Prof. T. Mikkonen, Univ. Lect. A.-P. Tuovinen

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	January 2, 2022	43 pages, 35 appendix pages

Tiivistelmä — Referat — Abstract

A US government agency estimated the national cost of inadequate software testing to be $60 billion annually, and that was 20 years ago. As the role of technology and software has been rapidly increasing worldwide for decades, it suffices to say that the worldwide fiscal effect of poor testing practices today is probably "quite a bit".

An increasing number of industry-focused survey studies on testing have been published worldwide in recent years, signalling an increased need to characterize the testing practices of the software development industry. These types of studies can help to guide future research efforts towards subjects that are meaningful to the industry, and provide practitioners with an opportunity to compare their own practice to those of their peers and recognize the main improvement areas.

As no secondary study devoted to these types of survey studies could be identified, the opportunity was seized to carry out a literature review was to find out what the data from these studies can tell us when aggregated. The precise topics focused on were the usage of test levels, test types, test design techniques, test tools and test automation.

Looking at these studies in aggregate tells us about some general trends: unit testing, functional testing and regression testing are popular everywhere, and also quite popular regardless of the surveyed population are performance testing and usability testing. The popularity of the other test levels and test types vary from survey to survey or region to region. Black-box techniques and experience-based techniques are more popular than white-box techniques. Exploratory testing, error guessing, use case testing and boundary value analysis are some of the most popular test design techniques. Much of the industry relies on manual testing over automated testing and/or have inadequately adopted the usage of testing tools.

ACM Computing Classification System (CCS)
Software and its engineering → Software creation and management → Software verification and validation

Avainsanat — Nyckelord — Keywords

software testing, industrial surveys

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software study track

Contents

1 Introduction

Software is everywhere, and the primary method of performing quality assurance on software is to conduct software testing. Software testing cannot guarantee the absence of defects, but it can help to prevent them [Int15], and efforts to do so are proven to be worthwhile time and again, sometimes in spectacular fashion. For instance, a defect in a trading algorithm cost a company $440 million in 30 minutes [Heu12] and another caused emergency services to be unavailable to over 10 million people for six hours [Fun14]. For a more grounded perspective, a likely outdated but often-cited estimate made in 2002 by the National Institute of Standards and Technology is that the cost of inadequate infrastructure for software testing in the US is $59.5 billion [Tas02]. Undoubtedly, this issue is not specific to the US and as the role of technology and software increases worldwide, so does the importance of software testing.

This notion is supported by the fact that an increasing number of survey studies that aim to find out about the testing practices of the software industry are being carried out worldwide. Characterizing the testing practices of the software development industry provides a reference point for software professionals in their practice, helps to guide future research efforts towards subjects that are relevant to the industry, and helps to gauge the adoption rate of techniques that are of interest in academia. Recognizing the regional weaknesses in software testing practices could also be used as a starting point for discussions on curricular shortcomings, or the need to shift in attitudes on the organizational level.

Surprisingly, no previous secondary study devoted to the subject of the testing practices of the software industry seems to exist. So, while conducting another survey on the topic would be a viable way to contribute, perhaps even more warranted is a secondary study on the subject. Therefore, the contribution of this thesis is a literature review on the topic to find out what looking at existing data in aggregate tells us about the testing practices of the industry. Specifically, the focus is on a set of topics that surveys on testing practices are likely to cover: the usage of test levels, test types, test design techniques, test tools and test automation.

The remainder of this thesis is structured as follows. First, background information is discussed in chapter 2. Chapter 3 presents the research methods, i.e., how the literature

review was carried out. The results are presented in chapter 4, followed by discussion in chapter 5. Finally, the conclusions are presented in chapter 6.

2 Background

This chapter consists of two parts: first, we establish relevant testing concepts and definitions in section 2.1 and then the research context is discussed in section 2.2.

2.1 Testing concepts and definitions

There are competing standards when it comes to basic software testing terminology: the Engineering Body of Knowledge (SWEBOK) [ISO15], the International Software Testing Qualifications Board Certified Tester Foundation Level Syllabus (ISTQB-CTFL) [Int18] and the international standard on software testing ISO/IEC/IEEE 29119 [ISO13]. It is therefore particularly important in this work to establish testing concepts and terminology in order to provide a cohesive structure when presenting the results and to avoid confusion. The definitions presented are not strictly based on any one of the aforementioned authoritative sources, but draw from each.

Testing is a group of activities performed to assess the quality of software. Testing always occurs on one of four **test levels**, namely

- **unit testing** (or component testing) if the target is an isolated software component,
- **integration testing** if the target is an isolated subset of a software system,
- **system testing** if the target is an entire software system, or
- **acceptance testing** if the target is an entire software system *and* the goal is to assess whether a complete software system is acceptable to the end-user [ISO13; ISO15; Int18].

Testing activities can be grouped into one or more **test types** based on their objectives. Oftentimes, the objective is to assess a specific quality characteristic of software. A fairly comprehensive, but non-exhaustive list of test types that assess quality characteristics from ISO/IEC/IEEE 29119 consists of

- **functional testing**, which evaluates how well the functionality of a product meets explicit and implicit requirements of the product under specific conditions,

- **compatibility testing**, which evaluates how well a group of components or systems interoperate on the same software environment or hardware,
- **usability testing**, which evaluates how easily and satisfactorily specific users, in a specific context, are able to use the product for its intended purpose,
- **reliability testing**, which evaluates how well a component or a system functions in specific conditions over a period of time,
- **security testing**, which evaluates how well access to data and data operations is restricted only to persons or systems with authorization to do so,
- **maintainability testing**, which evaluates how easily changes can be made to the product,
- **portability testing**, which evaluates how easily a component or system can be made available on other hardware, software or in other environments, and
- **performance testing**, which evaluates the performance of a product in relation to the resources used under specific conditions [ISO13].

Practitioners will often refer to the group of test types that focus on a specific quality characteristic excluding functional testing as **non-functional testing** [ISO13]. Some notable test types which are not tied down to a specific quality characteristic are:

- **regression testing**, to re-test already tested software to assess whether changes to software have had any undesirable effects,
- **user acceptance testing** (UAT), a form of acceptance testing carried out by the customer, to verify that the software meets their requirements,
- **alpha testing**, a form of acceptance testing feedback is obtained by allowing a limited number of users to use the software before it is released. Useful for off-the-shelf software, where UAT is not applicable [Int18].

As the objectives of testing are innumerable, so is the number of test types. Others may come up in the results but are likely not significant enough to warrant discussion. It should be noted that a test type can be applicable to more than one test level. For example, both functional testing and performance testing could be conducted on both the integration

and system test levels in one project. Acceptance testing, under the given definitions, could also be considered as a special kind of test type that only applies to the system test level. Defining test level and test types in such a way helps to accommodate studies that use the SWEBOK model, under which acceptance testing is likened to test types instead of test levels [ISO15]. Specific forms of acceptance testing, e.g. UAT and alpha testing are discussed as test types.

In testing, there is the matter of how test cases are derived. The techniques used for this purpose are called **test design techniques**. Test cases can be derived using specifications (e.g. requirements), structure (e.g. source code), or experience. Using specifications to identify test cases is called **black-box testing**, and using source code is called **white-box testing**. Definitions have been omitted for brevity; refer to SWEBOK for definitions [ISO15]. Table 2.1 lists various test design techniques.

Table 2.1: Non-exhaustive list of test design techniques.

White-box	**Black-box**	**Experience-based**
Control Flow	Equivalence partitioning	Ad hoc testing
Data Flow	Boundary value analysis (BVA)	Exploratory testing
Error guessing	Pairwise Testing	Error guessing
Mutation testing	Random testing	
Decision Tables		

The international standard on software testing ISO/IEC/IEEE 29119 uses three layers to describe software testing in the context of organizations and projects [ISO13]. First, there is the organizational layer, the context of which is made up of rules, regulations, standards and laws. In mature organizations, formalized testing policies and strategies may be produced based on these inputs. The organizational layer provides context, along with the project context, for the middle layer, which is the project layer. This layer may also include some formal planning and strategizing. It is also the layer where it is determined whether a sequential development model like the waterfall model or an iterative model like agile is used, among other things which influence day-to-day testing activities in the project. Thus, the project layer provides context for the final layer, the testing layer, which deals with the nuts and bolts of testing such as test levels, test types, test design techniques, test completion criteria and test automation.

It is likely a reasonable assumption that trends observed in the testing layer are often results of trends in the project layer. For instance, the transition towards agile practices and continuous integration has almost certainly increased the amount of regression testing conducted. However, there is no intention of exploring which factors have shaped testing practices in the context of organizations and projects; the goal is to observe industry trends and patterns of the software testing layer in isolation.

2.2 Research context

There are a few approaches to learning about the testing practices used in the software industry. One approach is to analyse publicly available data, such as job postings [Kas+21; CLR20; FS19] and public repositories [LSM20]. Using this approach allows for an efficient, automated data gathering process but can be limited by the data which is available and the lack of structure can cause complications. The more common alternative involves reaching out to members of the industry directly by means of a questionnaire survey, interviews, or a combination of the two. Broadly speaking, these types of studies can be categorized into qualitative and quantitative studies. Qualitative studies on the topic, usually case studies, can provide detailed insight on the testing process of a company or handful of companies, but their results are usually not generalizable to the software industry as a whole as the number of subjects is small. In order to answer questions such as "*how popular is the software testing practice X?*", perhaps the best tool for the job is industrial surveys.

With surveys, there is the question of geographical coverage. To provide a generalizable answer to a question about the popularity of a testing practice, sampling would need to be done on developers and testers all around the world. Such surveys can and have been done [IST16; IST18], but much more common are surveys targeting a specific country or region, no doubt because it is more practical to conduct surveys on a smaller scale. This limitation is, however, a common source of criticism for industrial surveys, and is a factor in why industrial surveys can be difficult to get published [TR13]. The aim of this thesis is to learn about the testing practices used in the industry by looking at industrial surveys in aggregate. One motivation for this study is that the limited geographical coverage of any individual paper can be mitigated to a degree by being reduced to a single reference point amongst many.

While a brief discussion on existing work is standard practice for any study, in one survey study on software testing practices an abridged literature review was carried out prior to

conducting the survey [Dia+17]. The aggregated results of their review that are most relevant to this study were that unit testing is the most popular testing level and that manual testing is performed more often than automated testing. To the best of my knowledge, this thesis is the first study that is primarily a literature review focusing on the software testing practices of the industry, questionnaire survey-based or otherwise.

Most research efforts in software testing are not industry-centric, instead, they are directed towards topics that by most indications are disconnected from the mainstream software industry and almost seem doomed to remain as such for perpetuity. A prime example of this is mutation testing. The method has been an increasingly prominent research area since its inception over 50 years ago [DS78; JH11], to the point of becoming one of the most actively researched software testing methods, with seven secondary studies covering the topic as of 2016 [GM16]. At a cursory glance, the number appears to have at least doubled in the last five years [Pap+19; SSL17; Piz+19; FPO18; GGS17; JSR17; ZPZ18; PV19; DA16]. Despite all the research, the issue of industry adoption continues to be offhandedly recognized even in the most recent research on the topic [Pet+21; BW21]. ISTQB, the de facto body of software testing certification makes no mention of mutation testing in any of their syllabuses according to the ISTQB Glossary [IST21a]. Mutation testing is not an isolated case in this regard. Search-based testing, combinatorial testing, symbolic execution and random testing have all been the subject of much research [GM16], yet they are only mentioned in passing by the material of the ISTQB, if at all. In fairness, ISTQB devotes a syllabus for model-based testing, which was covered by fifteen secondary studies as of 2016, more than any other testing method [Int15; GM16]. As illustrated by Figure 2.1, according to Google Trends, interest in model-based testing has nevertheless remained stagnant and on the same level as mutation testing for over 10 years. The points presented here are meant to be taken as indicators, not hard evidence. Industrial surveys, however, might be able to more reliably tell us about the adoption rate of these highly researched software testing practices.

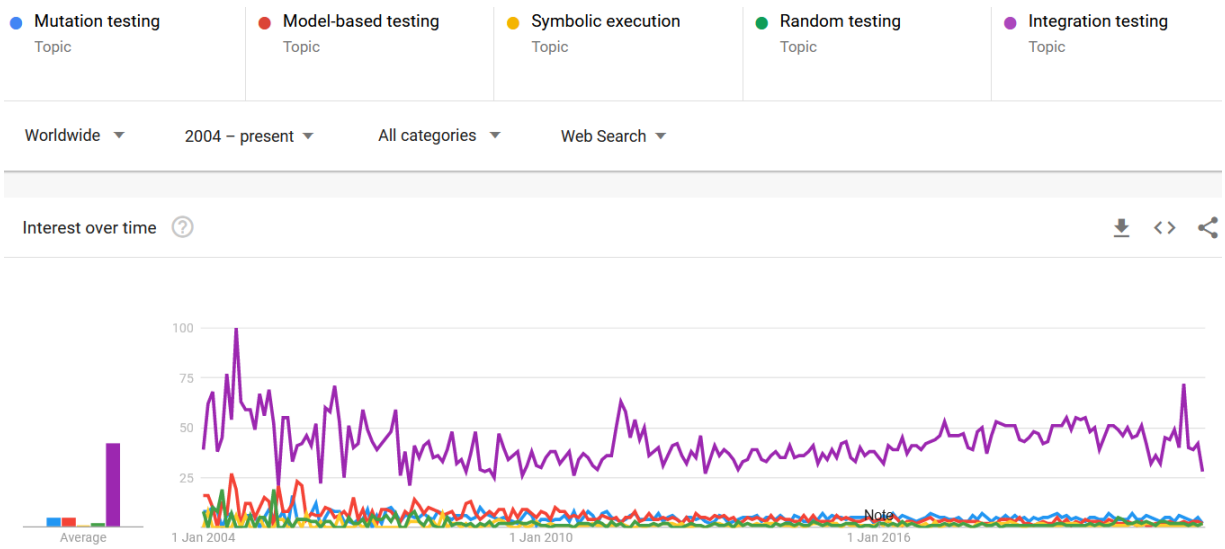

Figure 2.1: Interest over time on various software testing topics. Integration testing, which arguably is an industry staple, is included as a reference point. Data source: Google Trends (https://www.google.com/trends).

3 Research method

The research method used in this thesis is the (systematic) literature review. The steps that are taken largely draw from the guidelines presented in [Kee+07], although strict adherence has not been attempted, particularly, a quality assessment of the primary studies was not made. This field of research does not seem to be broad and mature enough for statistical analysis, an activity that would call for full adherence. Instead, a somewhat more lenient approach was taken, as this work is more in the realm of general, high-level observations of possible trends and patterns. More on this topic is available in the discussion on research validity (section 5.3).

Each section that follows corresponds to a consecutive step taken in the process conducting this study prior to synthesizing the results: formulating research questions (section 3.1), seeking out potential primary studies (section 3.2), filtering irrelevant primary studies (section 3.3), snowballing (section 3.4) and data extraction (section 3.5).

3.1 Research questions

The basic research question of this study is:

- RQ: What does descriptive quantitative data from questionnaire surveys tell us about the testing practices of the general software industry?

Reviewing all aspects of testing would be too laborious for a master's thesis; instead, a few key topics which are interconnected have been selected, and consequently, some more granular research questions are formulated: In the industry,

- (RQ1) How broadly are the test levels used, and what is their relative popularity?
- (RQ2) What are the most popular test types?
- (RQ3) What are the most popular test techniques?
- (RQ4) What is known about the usage of test tools/automation?

3.2 Search process

Primary studies were identified using the following electronic databases relevant to computer science:

- ACM Digital Library (Association for Computing Machinery)
- IEEE Xplore (Institute of Electrical and Electronics Engineers)
- Scopus

After a few preliminary rounds of the search process, the following search terms were settled upon:

- software AND
- testing AND
- practices AND
- survey AND
- industry OR industrial OR organization OR company OR practitioner OR professional

For each database, a search query was constructed in such a way that each group of keywords had to match either the title, abstract or keywords of a study. This is the default behaviour in IEEE Xplore and Scopus. The keywords have been constructed using the high-level research question for this study, and not the more specific ones, i.e., using keywords such as "tools", "automation" or "unit testing". The reasoning behind this is that based on early rounds of the search process, had this been done, it would be more likely for relevant literature to not be found as the terms are not necessarily used in the title, abstract or keywords of a survey study that broadly observes testing practices. While this led to a more laborious search process, it also provided some flexibility with regard to adjustments to the granular research questions.

The databases were queried on 8.11.2021 and the search queries used are presented in Table 3.1. The number of results for each database is presented in Table 3.2.

Table 3.1: The search queries used for each database.

Database	Search query
ACM	(Title:(software) OR Abstract:(software) OR Keyword:(software)) AND (Title:(testing) OR Abstract:(testing) OR Keyword:(testing)) AND (Title:(practices) OR Abstract:(practices) OR Keyword:(practices)) AND (Title:(survey) OR Abstract:(survey) OR Keyword:(survey)) AND (Title:(industry OR industrial OR organization OR company OR practitioner OR professional) OR Abstract:(industry OR industrial OR organization OR company OR practitioner OR professional) OR Keyword:(industry OR industrial OR organization OR company OR practitioner OR professional))
IEEE	software AND testing AND practices AND survey AND (industry OR industrial OR organization OR company OR practitioner OR professional)
Scopus	TITLE-ABS-KEY(software AND testing AND practices AND survey AND (industry OR industrial OR organization OR company OR practitioner OR professional)) AND (LIMIT-TO (SUBJAREA, "COMP"))

Table 3.2: Number of studies per database.

Database	Number of results
ACM	251
IEEE	186
Scopus	257
Total	694

3.3 Exclusion criteria

The following exclusion criteria were used to narrow down the set of papers to only the ones relevant to this research:

- EC1: Published before 2001
- EC2: Published in a language other than English

- EC3: Is not a survey study with a sample size of at least 20
- EC4: Has an insufficient focus on testing practices of the general software industry

Applying the exclusion criteria yielded a total of 16 papers. The intermediate number of results after removing duplicates and applying each exclusion criteria are presented in Table 3.3.

Table 3.3: The process of narrowing down the selection of papers.

Phase	Number of results
Initial	694
Removal of duplicates	495
Applying EC1	448
Applying EC2	445
Applying EC3 & EC4	16

3.4 Snowballing

Backwards snowballing is the process of checking the references of a set of seed papers for potential sources to be included in a review. In forwards snowballing, the referees are checked instead. Snowballing can be repeated iteratively on the additional sources until no new sources are found.

To improve the coverage of the review, both forwards and backwards snowballing was applied to the set of 16 papers. On the first iteration, two survey studies published in scientific venues which passed the exclusion criteria were identified. The snowballing process was terminated after a second iteration of snowballing, as no new papers were found. In addition, the backward snowballing process allowed for some grey literature to be included, specifically, surveys published by the ISTQB or its member boards. Fifteen such surveys are available on the ISTQB website [IST21b], of which nine passed the exclusion criteria. Not all of them were reachable via direct snowballing, but they were included nevertheless due to the high relevance. Table 3.4 summarizes the snowballing process.

Table 3.4: The snowballing process.

Set	Number of results
Initial papers	16
Scientific literature added via snowballing	2
ISTQB surveys added	9
Final number of surveys included in review	27

3.5 Data extraction

The following data was extracted for each paper:

- Study metadata
  - Title
  - Year
  - Author(s)
  - Reference
- Survey metadata
  - Year(s) conducted
  - Number of responses
  - Target population (individuals, organizations...)
  - Region/scope
- Quantitative findings pertaining to the usage of
  - test levels (RQ1)
  - test types (RQ2)
  - test design techniques (RQ3)
  - automation and tools (RQ4)

The extracted data is available in full in Appendix A. If otherwise pertinent data was not extracted for some reason, e.g., due to suspicion of faulty data or misreporting, it was noted with justifications. For Likert scale questions, the mode and median were extracted as they best measure the central tendency for Likert data [BB12]. During data extraction, several times an opportunity arose to convert Likert data into binary data with the intention of it comparing to data from other studies obtained using a yes/no or multiple-choice question. For instance, from a Likert scale question with options ranging from “never“ to “always“, one could infer “never“ to be equivalent to “no“, and the remaining options to be equivalent to “yes“. This was initially done in this study, but it was subsequently undone as it quickly became apparent the data obtained this way were consistently outliers, suggesting that the inference is not correct.

4 Results

In this chapter, the results of this study are presented. First, an overview of the studies and the respective surveys is presented in section 4.1. The remaining four sections of this chapter each correspond with the research question of this study, starting with test levels in section 4.2, followed by test types in section 4.3, test design techniques in section 4.4 and lastly automation and tools in section 4.5.

4.1 Overview of primary studies

Table 4.1 provides an overview of the studies included in this literature review. Something worth noting is the steady increase in the number of studies on the topic; even though 20 years' worth of literature is under examination, around half the studies have been published in just the last 5 years.

Table 4.1: Basic information about the studies included in the review.

Id	Year	Author(s)	Title	Reference
1	2004	Geras et al.	A survey of software testing practices in Alberta	[GSM04]
2	2004	Ng et al.	A preliminary survey on software testing practices in Australia	[Ng+04]
3	2010	Garousi et al.	A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009?	[GV10]
4	2010	Causevic et al.	An industrial survey on contemporary aspects of software testing	[CSP10]
5	2011	Haberl et al.	Software Test in Practice	[Hab+11]
6	2012	Lee et al.	Survey on software testing practices	[LKL12]
7	2012	Turkish Testing Board	Turkey Software Quality Report 2011-2012	[Tur12]
8	2013	Garousi and Zhi	A survey of software testing practices in Canada	[GZ13]
9	2013	Turkish Testing Board	Turkey Software Quality Report 2012-2013	[Tur13]
10	2014	Daka and Fraser	A survey on unit testing practices and problems	[DF14]
11	2014	Turkish Testing Board	Turkey Software Quality Report 2013-2014	[Tur14]
12	2015	Garousi et al.	A survey of software engineering practices in Turkey	[Gar+15a], [Gar+15b]
13	2016	ISTQB	ISTQB® Worldwide Software Testing Practices Survey 2015-16	[IST16]
14	2016	Turkish Testing Board	Turkey Software Quality Report 2015-2016	[Tur16]
15	2017	Dias-Neto et al.	Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay	[Dia+17]

16	2017	Kassab et al.	Software Testing: The State of the Practice	[KDL17]
17	2017	Felderer and Auer	Software quality assurance during implementation: Results of a survey in software houses from Germany, Austria and Switzerland	[FA17]
18	2018	Hynninen et al.	Software testing: Survey of the industry practices	[Hyn+18]
19	2018	Wang and Galster	Development processes and practices in a small but growing software industry: A practitioner survey in New Zealand	[WG18]
20	2018	ISTQB	ISTQB® Worldwide Software Testing Practices Survey 2017-18	[IST18]
21	2018	Turkish Testing Board	Turkey Software Quality Report 2017-2018	[Tur18]
22	2019	Quesada-López et al.	A survey of software testing practices in Costa Rica	[QHJ19]
23	2019	Turkish Testing Board	Turkey Software Quality Report 2018-2019	[Tur19]
24	2020	Latif et al.	A preliminary survey on software testing practices in Khyber PakhtunKhwa region of Pakistan	[LR20]
25	2020	Vukovic et al.	An empirical investigation of software testing methods and techniques in the province of Vojvodina	[Vuk+20]
26	2021	Junior et al.	Experiences and Practices in GUI Functional Testing: A Software Practitioners' View	[Jun+21]
27	2021	Carlos and Ibrahim	Practices in software testing in Cameroon challenges and perspectives	[MI21]

Table 4.2 provides some additional data about the survey conducted in each study. Most surveys were conducted around 1-2 years before the publication of the study. Around two-thirds of the studies target individual practitioners, and the rest target organizations and companies. The median number of responses is around 100.

Table 4.2: Basic information about the surveys.

Study id	Year(s) conducted	Responses analyzed	Target population	Region/scope
1	2002	60	Software organizations	Alberta, Canada
2	2002-03	65	Software organizations	Australia
3	2009	53	Software organizations	Alberta, Canada
4	2009	83	Software companies	Europe
5	2011	1623	Software professionals	Central Europe
6	Unknown	24	Software professionals	Major companies
7	2011-12	Unknown	Software professionals	Turkey
8	2010	246	Software practitioners	Canada
9	2012-13	Unknown	Software professionals	Turkey
10	2013	171	Software developers	Worldwide
11	2013-14	Unknown	Software professionals	Turkey
12	2013	202	Software practitioners	Turkey
13	2015	3200	Software professionals	Worldwide
14	2015-16	Unknown	Software professionals	Turkey
15	2015	150	Testing practitioners	Brazil & Uruguay
16	2015	167	Software professionals	USA
17	2015	57	Software houses	Central Europe
18	2017	33	Software organizations	Finland
19	2017-18	101	Software developers	New Zealand
20	2017-18	2000	Software professionals	Worldwide
21	2017-18	300	Software professionals	Turkey
22	2018	92	Software professionals	Costa Rica
23	2018-19	Unknown	Software professionals	Turkey
24	2018	70	Software organizations	Khyber PakhtunKhwa, Pakistan
25	Unknown	83/24	Software professionals/Software organizations	Vojvodina, Serbia
26	2021	222	Testing professionals	Brazil
27	Unknown	30	Software companies	Cameroon

Figure 4.1 visualizes the geographical coverage of the studies. The most covered regions are Turkey (7 studies), largely thanks to the surveys carried out by the Turkish Testing Board (TTB), and Canada (3 studies). The coverage from five studies is not included (or only partially included) because the geographical distribution was not reported. This includes two worldwide ISTQB studies, which both received responses from around 90 countries [13] [20].

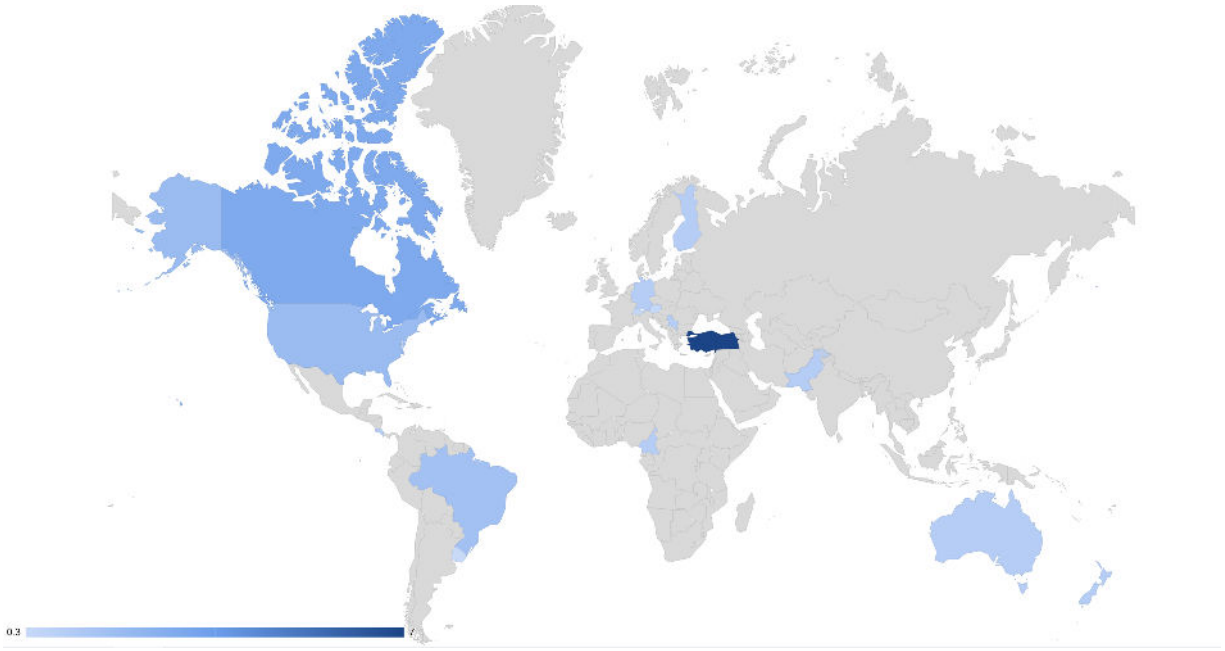

Figure 4.1: The geographical coverage of the primary studies.

4.2 Test levels

Each of the test levels tend to be used more than two-thirds of survey respondents, but the results are only consistent for unit testing; see table 4.3 for the raw data on the popularity of test levels, and table 4.4 for basic statistics about the results. It should be noted that no claims are being made about the statistical significance of this data; the quality of the underlying studies is not necessarily sufficient for meta-analysis, and the studies have not been weighted. The purpose of displaying statistics is only to aid in the digestion of the findings.

Table 4.3: The findings on the usage of the test levels in the reviewed studies. Whole numbers represent the percentage of respondents who indicated that they conduct testing on the test level. Fractions loosely represent the result of a Likert scale question (median value; 0 ('Never') is lowest).

Study id / **Test level**	1	3	4	8	12	15	16	17	19	22	27
Unit	92	96	5/7	80	2/4	76	72	68	73	79	73
Integration		37			3/4	82	67		48	81	77
System	92	97	6/7	78	3/4	88	81		26	83	27
Acceptance	80	88				83	72		45	88	27

Table 4.4: Basic statistics on the (non-Likert) findings concerning the popularity of the test levels.

Test type	**Min**	**Max**	**Mean**	**Median**	**Standard deviation**
Unit	68	96	79	76	9
Integration	37	82	65	72	19
System	26	97	72	82	28
Acceptance	27	88	69	80	23

The popularity of integration testing, system testing and acceptance testing test level can vary greatly depending on the region. They have each been a low outlier in at least one study: integration testing was by far the least popular test level in Alberta [3] and system testing was least popular by some margin in New Zealand [19]. System testing and acceptance testing were both clearly less popular than unit testing and integration testing in Cameroon [27].

The positions of survey respondents may have a non-negligible impact on the outcomes of survey studies on software testing practices such as test level usage. One study reported their figures on the popularity of test levels for respondents in management positions and technical respondents separately, with 72% of respondents in management positions responding affirmatively to using test levels while the corresponding figure for non-management respondents was 92% [1]. It was suggested the difference may be attributed to a lack of communication between technical and managerial project members, as managerial respondents may simply not be aware of the more technical testing levels such as unit testing being performed. To mitigate this issue, the values for whichever figure

was higher was reported in table 4.3, as they were presumably more accurate. However, many of the reviewed studies have both technical and managerial respondents, and may have been affected by this problem.

Even with a relatively high number of primary studies covering the topic, the geographical coverage of data on test level usage is lacking. No broad surveys (e.g. ISTQB surveys) have explored the topic, and for instance, not a single data point exists on the usage of system testing in the entire Eurasian mainland. The data that does exist is somewhat skewed towards North America (5/11 studies).

4.3 Test types

The test types which are consistently shown to be popular are, in decreasing order of popularity, functional testing, regression testing, performance testing and usability testing. See table 4.5 for the raw data on the popularity of test types, and table 4.6 for basic statistics on the most prominent test types. The relative popularity of the aforementioned test types is consistent across the studies, with the exception of study 27, where usability testing was more popular than regression testing and performance testing. This, combined with the fact that the variance in the results on these test types is quite low provides some confidence in accuracy of these results. Although security is regarded as prominent among survey designers, having the second-highest number of data points, its popularity in the industry varies greatly depending on the surveyed population. As for the remaining test types, less data is available, so the results should be interpreted with care. There are not very many data points for types of acceptance testing, only four for UAT and two for alpha testing, each hovering around 40-60%. In a few studies, stress testing was used by 60% of respondents, complementing the findings on performance testing. There are a dozen “Ility“-test types which do not appear to be in contention for being among the most popular test types.

Table 4.5: The findings on the usage of the test types in the reviewed studies. Types are sorted by the number of data points, descending. Whole numbers represent the percentage of respondents who indicated that they conduct testing on the test level. Fractions loosely represent the result of a Likert scale question (median value; 0 ('Never') is lowest). Only types reported on by two or more studies are included.

Study id / **Test type**	1	2	3	4	5	7	8	12	13	15	16	19	20	22	23	27
Performance				5/7	50	83	56	2/4	63	70	63		61	71	64	43
Security	44		13		26			2/4	39	70			45	71	33	43
Regression	72	69	68		80					83	63			79		53
Functional				6/7		98	78	3/4			91		83		92	
Usability	53		47			43			56				44		62	63
Scalability									15		19		16		8	3
UAT		48					56	3/4					66			
Stress	63		57					2/4			30					
Interoperability									19		21		15		12	
Reliability									31		41		22		26	
Recoverability									13				10		11	7
Accessibility									29				28		35	
Availability									26				20		28	
Maintainability									19				19		25	
Testability									28				21		34	
Operability									18				13		13	
Portability									11				11		13	
Supportability									8				7		8	
Extensibility									5				4		8	
Efficiency									26				19			
Alpha	40		50													

Table 4.6: Basic statistics on the (non-Likert) findings concerning the popularity of of the most prominent test types.

Test level	**Min**	**Max**	**Mean**	**Median**	**Standard deviation**
Functional	78	98	88	91	8
Regression	53	83	71	71	10
Performance	43	83	64	63	11
Usability	43	63	53	53	8
Security	13	71	43	43	19

As with test levels, the outcomes of results on the popularity of test types may be affected by the positions of the respondents. While technical respondents more commonly reported that test levels were used, managerial respondents were twice as likely to report that their

organization conducted usability testing [1]. Again, it has been presumed that whichever figure is higher is also more accurate, and again, the issue may have affected other studies.

There are some quirks when it comes to the usage of the term functional testing in the literature which sometimes gives pause when aggregating results. The problem is perhaps best explained with a quote from one of the primary studies, describing a phase in their survey design:

> The goal behind this phase in our survey design was to ensure that the terminology used in our survey was familiar to a reasonable ratio of the audience. This is since, unfortunately, the software testing terminology used in academia versus industry can sometimes be slightly different or even confusing, e.g., system testing and functional testing. [8]

In the study, the terms system testing and functional testing were used interchangeably, and sometimes "functional/system testing" was used. One study by the TTB reported on "functional/regression testing" [7], and another study reported on "functional black-box system testing" [4]. As the relatively high popularity of functional testing is a fairly uncontroversial finding, the general confusion around the term, and the tendency to have it attached to other qualifiers is not necessarily a problem when interpreting the results, but may be worth noting nevertheless.

4.4 Test design techniques

Before presenting the findings on the popularity of test design techniques, it should be noted that the results are somewhat less clear than those pertaining to test levels and test types. This is largely because the concept of test design techniques and the terminology used to describe test design techniques is less established that of test types and test levels. It is not uncommon for studies to report on test design techniques which are not recognized by SWEBOK [ISO15] or the ISTQB Glossary [IST21a], e.g. "symbolic analysis" [2]. As an example of inconsistency in terminology, it is probable that the following terms which have been used to describe or label data on the same technique: "Equivalence", "Equivalence classing", "Equivalence partitioning" [5] [11], "Category partitioning" [9] [12], "Equivalence partitioning" [20], "Equivalence class partitioning" [25] and "Partitioned analysis" [27]. It is worth considering whether the respondents to a particular survey have necessarily been aware

of the specific term used in the survey. Data on specific techniques has been grouped somewhat conservatively so that this can be accounted for, particularly for outliers.

The data on the popularity of test design techniques, presented in table 4.7, suggests that experience-based techniques and black-box techniques are more popular than white-box techniques. In only two instances did over 50% of respondents indicate that they use white-box test design techniques [5] [25], and in each instance black-box techniques and experience-based techniques were more popular when data was available. Not in a single study was a white-box technique most popular. It is difficult to tell from this data whether experience-based techniques or black-box techniques are more popular.

As for individual test design techniques, the data is quite scattered and there is not enough data on many techniques to give a definitive answer. The individual techniques which somewhat stand out as popular are error guessing, exploratory testing, use case testing and boundary value analysis.

Table 4.7: Summary of findings on the usage of the test design techniques from various studies. Whole numbers represent the percentage of respondents who indicated that they use the test design technique. Fractions loosely represent the result of a Likert scale question (median value; 0 ('Never') is lowest).

Study id	1	2	3	4	5	8	10	11	12	13	15	16	20	22	23	25	27
Experience-based					95	38			49								
Tester skill	91		91														
Error guessing					85			54		37			36		43		
Exploratory						5		49		66	81		67	69	42	2/4	
Exploratory (w/o objectives)					66												
Exploratory (w/ objectives)			33		33												
Fault attack with defect checklists					22												
Attacks								22		10			9		13		
Black-box		45		6/7	97							78				92	
Customer requirements	72		78														
Functionality testing					82												
Use case testing					77			76		71			73		69		
User story testing															61		
Model-based testing						15			22								
Checklist based								60		54			50		61		
Risks	43		20														
Cause effect graphs	3		10													24	
Out of range	52		47														
Boundary values	28		45														
Boundary value analysis					68	28		41	24				52		38	60	
Decision tables	10		8		33			32		22			29		26		
Equivalence partitioning	5		16		61	24		16	24	34			36		25	22	
Partitioned analysis																	13
Random testing					39												
State transition testing					38			24		21			21				
States	17		22														
Pairwise testing					15			10		13			13		9		
Combinatorial test techniques																49	
Classification tree method					15			11		7			6		15		
White-box		25			62	22						42				78	
Control flow graphs	5		10													45	
Data-flow analysis		28															
Mutation analysis/testing		5				29	4/6										0/5
Symbolic analysis		0															
Statement coverage					42			21		18			22		16	48	
Decision/branch coverage					36			21					25		22	53	
Condition coverage					23												
Path coverage					22												
Multiple condition coverage					13												
Condition determination coverage					8												
Loop coverage					8											55	
Source code analysis									33								
Code complexity analysis																36	

4.5 Automation and tools

This section is split into multiple subsections, as the goal was not only to aggregate the findings on the popularity of automation and tools, but also any other aspects of automation and tools for which quantifiable data exists. There is a lot of data available on the subject of automation and tools, with three in every four of the reviewed studies visiting the subject in some way. This, along with the fact that no two independent studies seem to approach the subject in the same way in terms of framing, scoping, granularity or terminology makes it quite challenging to aggregate and categorize results. Being too strict about only grouping data which is perfectly compatible leads to a poor structure in presenting results, and does not allow for the benefits of analysing results in aggregate. Not being strict enough risks comparing totally incompatible data, apples to oranges so to speak, risking outright invalid analysis. An attempt has been made to find an appropriate balance in this regard, but it may be advisable to refer to the extracted data in appendix A or the primary studies themselves if the reader particularly interested in granular findings than the broad overview given here.

First, findings on the overall usage of automation and tools in testing, as well as the activities during which automation and/or tools are applied are presented in section 4.5.1. Then, the results on the usage of specific tools are discussed in section 4.5.2. Finally, the matter of how testing tools are acquired is discussed in section 4.5.3.

4.5.1 Usage of automation and tools

Studies generally indicate that up two two-thirds of the industry automates testing, as shown in table 4.8, however, studies also consistently find that the actual number of test cases which are automated is less than 20%, as shown in table 4.9. In other words, it would appear that a large chunk of the industry uses no automation at all, and those who do, generally still mostly rely on manual testing.

Table 4.8: Percentage of respondents using tools or automation in general (second row) to fully or partially conduct various testing activities (e.g. unit testing, regression testing, static analysis or test planning).

Study id	1	2	3	6	7	9	13	14	15	16	17	18	19	20	21	22	23	26	27
Automation in general							67					67	62	70	57				40
Test levels																			
Unit	32		65	26		34	43					57		44			38	12	
Integration	11		27	34								46						28	
System	22		45	47								27							
Acceptance				25															
Test types																			
Regression	26		45															1	
Functional/Regression					58														
Stress	29		20																
Performance					44	43	55	66				49		51			50	21	
Security												24						1	
Static/dynamic analysis																			
Static analysis						16	26				37			28	18		24		
Dynamic analysis							16							17	16		11		
Other activities																			
Test execution		54				36	70		80	34				70		71	49		
Test analysis and reporting		42																	
Test reporting				26															
Test management						33	65							65	70		58		
Test case management					40							46							
Test planning/management		26																	
Test planning				48															
Test estimation				48					74						62				
Test design						13	50					15		44			37		
Generating test cases/scripts		31							71							58			
Defect management				90															
Bug tracking				76	64		81		90		84	72		81	52	82	59		
Requirements management					24														
Requirements traceability							47							46			31		
Code profiling						16													

Table 4.9: A summary of findings pertaining to the usage of test automation of tools which were not applicable to table 4.8.

Unit testing is the most automated test level, with most respondents automating 70-85% of their unit tests. Most respondents automate 15-30% of integration and system tests. Least automated is acceptance testing [5].
Testers are more likely to use tools than developers. Most (36.6%) testers use test tools in all projects, while developers typically use them in most (30.0%) projects or some (28.2%) projects. [5]
Respondents typically automate fewer than 20% of their test cases, on average [8] [13] [20] [23].
Respondents 'Frequently' (mode and median) automated test generation, most commonly for finding crashes and undeclared exceptions (41%), exercising assertions in code (38%) or exercising parameterised unit tests (38%). [5]
Respondents 'Sometimes' (median) used automated tools for both code inspections and static code analysis. Respondents 'Seldom' (mode and median) used test automation [12].
Over 50% of respondents do not use any automation when conducting GUI tests, and less than 10% use automation exclusively [26].

The number of respondents using automation/tools when conducting any of the test levels is typically below 50%. With the exception of unit testing, results on the usage tools/automation per test-level is fairly scarce and inconsistent so not many other reliable observations can be made from the data. As for test types, only performance testing stands out as typically being automated by around 50% of respondents across many studies. Between static analysis and dynamic analysis, both activities which require tools, the former is more popular but neither is usually used by over one-third of respondents. Bug tracking stands out as being the activity for which tools are used the most, typically by around two-thirds of respondents or more. Also somewhat popular is the usage of tools/automation for test management and test execution. For other activities, the data is too scarce to make meaningful inferences.

4.5.2 Usage of specific tools for testing activities

Results on the popularity of testing tools are scarce, and almost any statement made on the topic is only backed by an individual data point, or can be met with counter-evidence. Two studies conducted in Canada found the unit testing frameworks **JUnit** (for Java) and **NUnit** (for C#) to be the most popular testing tools, with more than half of respondents using one or the other [3] [8]. The only additional data on these tools from these studies comes from a 2021 study in Cameroon, which found that of those who automate testing, only 13% of respondents used JUnit, which is particularly surprising considering that two-thirds of the respondents used Java and 74% of respondents conduct unit testing [27]. This same study instead found **Selenium**, a test automation tool for web applications, to be the most popular tool, used by around 40% of the industry, a finding backed up by another recent study focusing on GUI testing [26]. This study found **IBM rational** testing tools to only be used by 1% of respondents, however, in two studies earlier studies it was used by a respectable chunk of respondents, 20% [3] [24]. Other highly-scoring (used by 10% of respondents or more) testing tools for which only an individual data point is available are **Appium** (20.95%) [26], FitNesse (17%) [3], **Sonarqube** (15%) and **findbugs** (10%) [27]. The study on GUI testing queried respondents about the tools they used for reporting bugs. Kanban (not a specific tool) scored highest, followed by **Jira** and **Slack**.

The lack of data on the topic of specific tool usage may be a consequence of the difficulty of designing a question on the topic of specific test tools, as a lot of tools exist and some effort has to be put into enumerating a reasonable set of options. One of the reviewed studies avoided this issue by using an open-ended question, asking for specific names of testing tools, and then simply reporting the raw responses [4]. Quantifiable inferences about the popularity of tools drawn from this data would likely be inaccurate because respondents may have trouble remembering all the tools used or their names, or understanding the question, as illustrated by some of the responses:

- “Two tools named something with "coverage" and "perform" (cannot remember the company behind)“
- “change control, bug tracking, test case management, etc., etc.“
- “Proprietary in most cases.“

4.5.3 How testing tools are acquired

The little data that is available on the subject of how testing tools are acquired suggest that both commercial and open-source tools are commonly used, and that developing custom tools is not uncommon, while out-sourcing the development of testing tools is. 2004 study in Australia found that the most common way of acquiring testing tools was to purchase existing commercial products (68%). Some developed their testing tools in-house (14%) and one respondent out-sourced the development of their testing tools (2%) [2]. The study makes no mention of how the remaining respondents (16%) acquired tools, although later studies would suggest that they were probably using open-source tools. A 2010 study in Alberta noted that both commercial and open-source tools are common; looking at the raw data for individual tools would suggest that open-source tools were more commonly used among respondents [3]. A 2020 study in Pakistan found that using open-source tools was slightly more common than using commercial tools (36% vs. 33%), and that one in four respondents developed tools for testing in-house [24]. Another study reported that 17% of respondents develop custom testing tools [27].

5 Discussion

This chapter begins with an analysis of the results (section 5.1), followed by revisiting the research questions (section 5.2), then a discussion on research validity (section 5.3), and ends with some thoughts on future work (section 5.4).

5.1 Analysis

Encouragingly, many of the surveyed populations have widely adopted most of the test levels, however, the comparatively low adoption rates of certain test levels in certain regions, e.g., integration testing in Alberta [3], system testing in New Zealand [19] and system testing and acceptance testing Cameroon [27] is puzzling. A quick investigation did not reveal any glaring methodological issues in the studies which could explain the outliers, except perhaps the low sample size (30) of the last study; and even in this case, there is not much evidence against the phenomenon being real, considering that all of the other results the popularity of system testing and acceptance testing come from outside the Afro-Eurasian mainland, so there is not much to compare the result to. A deeper investigation might reveal that the source of these outliers are some forms of bias in the studies, however, entertaining the possibility that the reported phenomenons are real, it is surprising how the impact of regional culture could be so great as to make one region reliant on one test level while overlooking another, and have the situation reversed in another region.

It stands to reason that the primary concern when developing software is that its functionality works as intended, so the result that functional testing is most popular likely does not come as a surprise to anyone. The popularity of regression testing speaks to that the industry mostly recognizes the high risk of defects being introduced when making changes to software. Then again, in many surveys, some one-third of respondents did not conduct regression testing. Either this means that the surveyed populations are creating a lot of software that is never updated ("shelfware"), or large portions of these industries are unnecessarily vulnerable to introducing defects when changing software. If the latter is the case, then there is room for improvements to be made in the general know-how of practitioners (i.e. by means of better education/training), and/or the resources allocated to testing (i.e. by means of shifting organizational attitudes) in many regions. The fact that

around half or more of the industry performs performance testing and usability testing is an encouraging result, particularly considering that many of those who do not probably aren't exposed to performance or usability issues due to the nature of the software in the first place. It is somewhat interesting how the relative popularity of the previously discussed test types does not seem heavily dependent on the surveyed population while for security testing and some of the test levels it does.

It is not too surprising that experience-based techniques and black-box techniques are more popular than white-box techniques, because white-box techniques such as those based on code coverage are only available to developers, and using them requires adopting specialized tools. Experience-based techniques and black-box techniques can be conducted by people without programming experience and typically do not require using specialized tools, so the threshold for adopting them is low.

Considering the ease with which unit tests can be written using open-source libraries like JUnit in a modern development environment, it is somewhat surprising that many still rely on manual testing, even on the unit test level. Perhaps it is worth considering what the automation of a testing activity like unit testing really means. It is clear that in a theoretical project where unit tests are created automatically and executed automatically (e.g. using continuous integration), unit testing has been automated. It is also quite clear that in another theoretical project, where a custom-made utility that requires manual inputs and observation of outputs is used to test a software component, unit testing is performed manually. However, there is a whole lot of ground between the two extremes: what about a theoretical project where tools are used to generate test cases automatically, which are then implemented manually using a unit testing framework that automates the comparison of expected outputs and actual outputs, and the automatic execution of all unit tests can be triggered manually? It may even be conceivable that the threshold for responding affirmatively to a question about the automation of unit testing is increasing over time, e.g., due progress is made in tools for automatic unit test generation. This is all to say that questions on the topic of automation are often too vague to really know what the underlying practices look like. Perhaps replacing questions about automation with questions on the usage of specific tools would be more beneficial.

5.2 Research questions revisited

Broadly speaking, all test levels are broadly used (RQ1). More specifically, the popularity of the test levels depends on the surveyed population, with the exception of unit testing which is consistently used by over two-thirds of respondents. The other test levels tend to be as popular or even more popular than unit testing, however, in some regions one or two test levels are used by as little as one-quarter of respondents, and there isn't a clear pattern as to which of the latter three test level ends up being a low outlier. As for the relative popularity of the test levels, the answer depends on surveyed population, and no generalizable answer can be given. For instance, even though the mean result for unit testing is fourteen percent higher than that of integration testing and the median result is higher as well, more times than not, integration testing was the more popular test level of the two.

The most popular test types in decreasing order of popularity are functional testing, regression testing, performance testing and usability testing (RQ2). Less consistently popular is security testing.

Results on the popularity of test design techniques are less clear, however, it is apparent that black-box test design techniques and experience-based test design techniques are more popular than white-box test design techniques (RQ3). Exploratory testing, error guessing, use case testing and boundary value analysis are some of the most popular test design techniques. Other techniques may be quite popular as well, but if this is the case, the fact has been obscured by the lack of conceptual and terminological consistency across the primary studies.

Typically, around two-thirds of survey respondents use tools to automate testing activities, however, fewer than 20% of test cases are automated (RQ4). None of the test levels stands out in terms of the level of automation; automation tools are typically used less then 50% of the time for each test level. Tools are quite often used for performance testing, test execution and bug tracking. Specific test tools which stand out as popular are NUnit, JUnit, selenium and IBM rational, although not much data exists on the topic as one might hope. Both commercial and open-source tools are common, and usually a small percentage of respondents develop their own tools.

These are only some of the things that descriptive data from questionnaire surveys can tell us (RQ). As many important aspects of testing were not explored in this study, it cannot be said that the overarching research question was answered satisfactorily. Examples of

such topics are testing metrics, testing standards, maturity models, testing training, static testing, test data, test environments, and organizational aspects of testing such as testing budget and the ratio of testers to developers.

5.3 Research validity

Assessing the quality of primary studies is a critical part of conducting a systematic literature review as it allows for studies of poor quality to be excluded via the exclusion criteria and it allows for quality to be taken into account when interpreting results [Kee+07]. No formal quality assessment was not conducted in this study, partially due to time constraints, but partially also because it was not deemed strictly necessary as the purpose of this study was not statistical analysis. The inferences made were made conservatively, only when there was plenty of supporting evidence and little or no counter-evidence. Even so, the validity of the inferences made in this study are to some degree threatened by a myriad of validity concerns of the primary studies themselves, such as poor sample sizes, sampling methods, and various bias-inducing pitfalls in questionnaire design. Setting a minimum threshold of 20 for the sample size in the exclusion criteria only somewhat mitigate this issue. A higher threshold was not used because high-quality studies with large sample sizes in this research area are few and far between.

It is advisable for a single researcher conducting a systematic literature review to re-evaluate the application of inclusion/exclusion criteria on a sample of already evaluated primary studies to ensure the consistency of their decisions [Kee+07]. No such measures were taken, so any mistakes made during the application of inclusion/exclusion criteria, or other error-prone phases of conducting this research for that matter, may have gone overlooked.

There are other threats to the validity of this study that could not so easily have been mitigated, such as the indeterminate effect of respondent demographics on the outcomes of the reviewed studies. For instance, here is a quote from one of the reviewed studies:

> The relatively high number of management respondents is important to consider when interpreting the results of the survey. An answer that indicates that the organization does not perform a certain testing level, for example, may mean either that the organization truly does not perform that testing

> level, or that is just not aware of the testing that goes on. The latter possibility is entirely reasonable, especially in cases of detailed testing such as unit testing [1].

The effect of demographic dimensions such as respondent position, years of experience, company size and industry sector can be speculated on to help understand outliers' data. However, their effects can not truly be accounted for without rigorous statistical analysis, something that is not possible for the vast majority of the reviewed studies. A particularly cumbersome issue is that survey participants can have various degrees of understanding of testing concepts and terminology. One study reported that 43.4% of respondents carry out usability testing, but noted respondents are likely confusing functional tests with usability tests and erroneously selecting the option, and that the actual percentage is much less [7]. In another survey, 50% of respondents claimed to be using tools to automate test generation, however, attaching an auxiliary open-ended question in a later version of the survey revealed that many respondents were confusing test generation tools with other testing concepts and even tools that are unrelated to testing [10]. These are only individual cases where researchers directly addressed the issue; the degree to which this has affected the outcomes of any of the reviewed studies is virtually unknowable. Some studies attempted to mitigate this issue with measures such as including definitions in the questionnaire, using a qualifying question to weed out respondents who lack a basic understanding of software testing [10], or by referring respondents to ISTQB glossary for explanations of terms. Besides respondent demographics and understanding of testing concepts, there is also an indeterminate list of factors regarding how the reviewed studies were carried out which may have skewed results, such as the lengths of the questionnaires and whether incomplete answers were included or discarded.

The exact figures reported from survey studies on software testing practices should be taken with a grain of salt regardless of the methodological soundness of the measurement, as people who want to take surveys on software testing are more likely to be interested in software testing than those who don't, and can therefore reasonably be assumed to be more likely to have adopted good testing practices than a randomly selected member of a sample. The degree to which this has affected the outcomes of the reviewed studies, and consequently, the outcomes of this study, is not easily known. It is probably a reasonable assumption that this form of bias is unlikely to have affected the relative popularity of testing practices, however.

It is possible that data has been misinterpreted during extraction. Most commonly, a

reported percentage in the reviewed studies represents a fraction of all respondents. However, sometimes studies report figures which only applies to a subset of respondents. For instance, a study might report on the percentage of respondents who automate unit testing *of those who conduct unit testing*. Naturally, comparing figures of one type with figures of the other type is not valid. For studies with high-quality reporting, the issue could easily be identified and adjusted for when encountered. In other cases, it is possible for misinterpretations to have occurred. This issue may also have been mitigated by conducting a quality assessment where shortcomings in reporting were taken into account.

5.4 Future work

Although no formal quality assessment was conducted in this work, there is an apparent need for more high-quality survey studies using methodological rigour including proper sampling methods, well-designed questionnaires and high-quality reporting to strengthen this research area. More geographical coverage from primary studies is needed as well, as there are no wide-scale survey studies written in English focusing on the software testing practices of a number of major software economies such as Russia, China, and India. Future literature surveys on the subject could verify the need for high-quality studies in this research area by conducting a quality assessment. There is also still uncovered ground for similar secondary studies to cover topics such as testing metrics, test completion criteria, and organizational aspects of testing.

6 Conclusions

As the importance of software grows, so does the importance of the primary method of conducting quality assurance on software, software testing. This notion is supported by the fact that many survey studies which aim to characterize the testing practices of the software industry have been published up in recent years.

The aim of this thesis was to aggregate results from these types of survey studies by means of a literature survey, with a focus on the usage of test levels, test types, test design techniques and test automation/tools.

The findings are that test levels are generally used by over two-thirds of surveyed populations but with the exception of unit testing, results can greatly vary widely depending on the surveyed population. The most used test types in decreasing order are functional testing, regression testing, performance testing and usability testing, and less consistently security testing. Experience-based techniques such as exploratory testing and error guessing and black-box techniques such as use case testing and boundary value analysis are more popular than white-box techniques. A lot of scattered data exists on the topic of automation/tools, but the principal finding is that the usage of automation is quite common, however, typically only a small fraction of test cases is automated, so overall the industry relies heavily on manual testing.

Bibliography

[BB12] H. N. Boone Jr. and D. A. Boone. "Analyzing Likert data". In: *Journal of Extension* 50.2 (2012). Publisher: JOE.

[BW21] M. Betka and S. Wagner. "Extreme mutation testing in practice: An industrial case study". In: *Proceedings - 2021 IEEE/ACM International Conference on Automation of Software Test, AST 2021*. IEEE, 2021, pp. 113–116.

[CLR20] M. Cerioli, M. Leotta, and F. Ricca. "What 5 million job advertisements tell us about testing: A preliminary empirical investigation". In: *Proceedings of the ACM Symposium on Applied Computing*. ACM, 2020, pp. 1586–1594.

[CSP10] A. Causevic, D. Sundmark, and S. Punnekkat. "An industrial survey on contemporary aspects of software testing". In: *ICST 2010 - 3rd International Conference on Software Testing, Verification and Validation*. IEEE, 2010, pp. 393–401.

[DA16] M. Dave and R. Agrawal. *Mutation testing and test data generation approaches: A review*. Vol. 628 CCIS. Communications in Computer and Information Science. Springer, 2016, pp. 373–382.

[DF14] E. Daka and G. Fraser. "A survey on unit testing practices and problems". In: *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*. IEEE, 2014, pp. 201–211.

[Dia+17] A. C. Dias-Neto, S. Matalonga, M. Solari, G. Robiolo, and G. H. Travassos. "Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay". In: *Software Quality Journal* 25.4 (2017). Publisher: Springer, pp. 1145–1183.

[DS78] R. A. DeMillo and F. G. Sayward. "Hints on test data selection: Help for the practicing programmer". In: *Computer* 11.4 (1978). Publisher: IEEE, pp. 34–41.

[FA17] M. Felderer and F. Auer. "Software quality assurance during implementation: Results of a survey in software houses from Germany, Austria and Switzerland". In: *Lecture Notes in Business Information Processing* 269 (2017). Publisher: Springer, pp. 87–102.

[FPO18] F. C. Ferrari, A. V. Pizzoleto, and J. Offutt. “A systematic review of cost reduction techniques for mutation testing: Preliminary results”. In: *Proceedings - 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops, ICSTW 2018*. IEEE, 2018, pp. 1–10.

[FS19] R. Florea and V. Stray. “A Global View on the Hard Skills and Testing Tools in Software Testing”. In: *Proceedings - 2019 ACM/IEEE 14th International Conference on Global Software Engineering, ICGSE 2019*. IEEE, 2019, pp. 143–151.

[Fun14] B. Fung. *How A dumb software glitch kept thousands from reaching 911*. Oct. 2014. URL: https://www.washingtonpost.com/news/the-switch/wp/2014/10/20/how-a-dumb-software-glitch-kept-6600-calls-from-getting-to-911/ (visited on 12/13/2021).

[Gar+15a] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs. “A survey of software engineering practices in Turkey”. In: *Journal of Systems and Software* 108 (2015). Publisher: Elsevier, pp. 148–177.

[Gar+15b] V. Garousi, A. Coşkunçay, A. Betin-Can, and O. Demirörs. *A survey of software engineering practices in Turkey (Technical report-Extended version)*. Tech. rep. 2015.

[GGS17] A. S. Ghiduk, M. R. Girgis, and M. H. Shehata. “Higher order mutation testing: A Systematic Literature Review”. In: *Computer Science Review* 25 (2017). Publisher: Elsevier, pp. 29–48.

[GM16] V. Garousi and M. V. Mäntylä. “A systematic literature review of literature reviews in software testing”. In: *Information and Software Technology* 80 (2016). Publisher: Elsevier, pp. 195–216.

[GSM04] A. Geras, M. Smith, and J. Miller. “A survey of software testing practices in Alberta”. In: *Canadian Journal of Electrical and Computer Engineering* 29.3 (2004). Publisher: IEE, pp. 183–191.

[GV10] V. Garousi and T. Varma. “A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009?” In: *Journal of Systems and Software* 83.11 (2010). Publisher: Elsevier, pp. 2251–2262.

[GZ13] V. Garousi and J. Zhi. "A survey of software testing practices in Canada". In: *Journal of Systems and Software* 86.5 (2013). Publisher: Elsevier, pp. 1354–1376.

[Hab+11] P. Haberl, A. Spillner, K. Vosseberg, and M. Winter. *Software Test in Practice*. 2011. URL: https://www.istqb.org/documents/Survey_GTB.pdf (visited on 12/30/2021).

[Heu12] M. Heusser. *Software testing lessons learned from Knight Capital Fiasco*. Aug. 2012. URL: https://www.cio.com/article/2393212/software-testing-lessons-learned-from-knight-capital-fiasco.html (visited on 12/13/2021).

[Hyn+18] T. Hynninen, J. Kasurinen, A. Knutas, and O. Taipale. "Software testing: Survey of the industry practices". In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*. IEEE, 2018, pp. 1449–1454.

[Int15] International Software Testing Qualifications Board. *Certified Model-Based Tester Syllabus - Version 2015*. 2015. URL: https://www.istqb.org/downloads/send/6-model-based-tester-extension-documents/46-istqb-ctfl-mbt-syllabus.html (visited on 12/30/2021).

[Int18] International Software Testing Qualifications Board. *Certified Tester Foundation Level Syllabus Version 2018 V3.1*. 2018. URL: https://www.istqb.org/downloads/send/2-foundation-level-documents/281-istqb-ctfl-syllabus-2018-v3-1.html (visited on 12/30/2021).

[ISO13] ISO/IEC/IEEE 29119-1:2013. *Software and systems engineering — Software testing — Part 1: Concepts and definitions*. International Standard. International Organization for Standardization, Sept. 2013.

[ISO15] ISO/IEC TR 19759:2015. *Software Engineering — Guide to the software engineering body of knowledge (SWEBOK)*. International Standard. International Organization for Standardization, Oct. 2015.

[IST16] ISTQB. *ISTQB® Worldwide Software Testing Practices Survey 2015-16*. 2016. URL: https://www.istqb.org/documents/ISTQB%5C%5FWorldwide%5C%5FSoftware%5C%5FTesting%5C%5FPractices%5C%5FReport.pdf (visited on 11/25/2021).

[IST18] ISTQB. *ISTQB® Worldwide Software Testing Practices Survey 2017-18*. 2018. URL: https://www.istqb.org/documents/ISTQB%202017-18%5C%5FRevised.pdf (visited on 11/25/2021).

[IST21a] ISTQB. *ISTQB Glossary*. 2021. URL: https://glossary.istqb.org/en/search/mutation%20testing (visited on 10/10/2021).

[IST21b] ISTQB. *ISTQB Surveys*. 2021. URL: https://www.istqb.org/references/surveys.html (visited on 11/24/2021).

[JH11] Y. Jia and M. Harman. "An analysis and survey of the development of mutation testing". In: *IEEE Transactions on Software Engineering* 37.5 (2011). Publisher: IEEE, pp. 649–678.

[JSR17] N. Jatana, B. Suri, and S. Rani. "Systematic Literature Review on Search based mutation testing". In: *E-Informatica Software Engineering Journal* 11.1 (2017). Publisher: Biblioteka Nauki, pp. 59–76.

[Jun+21] N. Junior, H. Costa, L. Karita, I. MacHado, and L. Soares. "Experiences and Practices in GUI Functional Testing: A Software Practitioners' View". In: *ACM International Conference Proceeding Series*. ACM, 2021, pp. 195–204.

[Kas+21] M. Kassab, P. Laplante, J. Defranco, V. V. G. Neto, and G. Destefanis. "Exploring the Profiles of Software Testing Jobs in the United States". In: *IEEE Access* 9 (2021). Publisher: IEEE, pp. 68905–68916.

[KDL17] M. Kassab, J. F. Defranco, and P. A. Laplante. "Software Testing: The State of the Practice". In: *IEEE Software* 34.5 (2017). Publisher: IEEE, pp. 46–52.

[Kee+07] S. Keele et al. *Guidelines for performing systematic literature reviews in software engineering*. Tech. rep. Citeseer, 2007.

[LKL12] J. Lee, S. Kang, and D. Lee. "Survey on software testing practices". In: *IET Software* 6.3 (2012). Publisher: IET, pp. 275–282.

[LR20] B. Latif and T. Rana. "A preliminary survey on software testing practices in Khyber PakhtunKhwa region of Pakistan". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 28.1 (2020). Publisher: The Scientific and Technological Research Council of Turkey, pp. 575–589.

[LSM20] J.-W. Lin, N. Salehnamadi, and S. Malek. "Test Automation in Open-Source Android Apps: A Large-Scale Empirical Study". In: *Proceedings - 2020 35th IEEE/ACM International Conference on Automated Software Engineering, ASE 2020*. IEEE, 2020, pp. 1078–1089.

[MI21] T. Maxime Carlos and M. Ibrahim. “Practices in software testing in Cameroon challenges and perspectives”. In: *Electronic Journal of Information Systems in Developing Countries* 87.3 (2021). Publisher: Wiley Online Library, e12165.

[Ng+04] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen. “A preliminary survey on software testing practices in Australia”. In: *Proceedings of the Australian Software Engineering Conference, ASWEC*. Vol. 2004. IEEE, 2004, pp. 116–125.

[Pap+19] M. Papadakis, M. Kintis, J. Zhang, Y. Jia, Y. L. Traon, and M. Harman. *Mutation Testing Advances: An Analysis and Survey*. Vol. 112. Advances in Computers. Elsevier, 2019, pp. 275–378.

[Pet+21] G. Petrovic, M. Ivankovic, G. Fraser, and R. Just. “Practical Mutation Testing at Scale: A view from Google”. In: *IEEE Transactions on Software Engineering* (2021). Publisher: IEEE.

[Piz+19] A. V. Pizzoleto, F. C. Ferrari, J. Offutt, L. Fernandes, and M. Ribeiro. “A systematic literature review of techniques and metrics to reduce the cost of mutation testing”. In: *Journal of Systems and Software* 157 (2019). Publisher: Elsevier.

[PV19] J. A. D. Prado Lima and S. R. Vergilio. “A systematic mapping study on higher order mutation testing”. In: *Journal of Systems and Software* 154 (2019). Publisher: Elsevier, pp. 92–109.

[QHJ19] C. Quesada-López, E. Hernandez-Agüero, and M. Jenkins. “A survey of software testing practices in Costa Rica”. In: *XXII Ibero-American Conference on Software Engineering, CIbSE 2019* (2019). Publisher: Ibero-American Conference on Software Engineering, pp. 503–516.

[SSL17] R. A. Silva, S. D. R. Senger de Souza, and P. S. Lopes de Souza. “A systematic review on search based mutation testing”. In: *Information and Software Technology* 81 (2017). Publisher: Elsevier, pp. 19–35.

[Tas02] G. Tassey. *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Tech. rep. National Institute of Standards and Technology, May 2002.

[TR13] M. Torchiano and F. Ricca. “Six reasons for rejecting an industrial survey paper”. In: *2013 1st International Workshop on Conducting Empirical Studies in Industry, CESI 2013 - Proceedings*. IEEE, 2013, pp. 21–26.

[Tur12] Turkish Testing Board. *Turkey Software Quality Report 2011-2012*. 2012. URL: https://www.istqb.org/documents/Survey%5C%5FTTB%5C%5F2011.pdf (visited on 11/24/2021).

[Tur13] Turkish Testing Board. *Turkey Software Quality Report 2012-2013*. 2013. URL: https://www.istqb.org/documents/Survey%5C%5FTTB.pdf (visited on 11/24/2021).

[Tur14] Turkish Testing Board. *Turkey Software Quality Report 2013-2014*. 2014. URL: https://www.istqb.org/documents/Turkey%20Software%20Quality%20Report%202013-2014.pdf (visited on 11/24/2021).

[Tur16] Turkish Testing Board. *Turkey Software Quality Report 2015-2016*. 2016. URL: https://www.istqb.org/documents/Turkey%20Software%20Quality%20Report%202015-2016.pdf (visited on 11/25/2021).

[Tur18] Turkish Testing Board. *Turkey Software Quality Report 2017-2018*. 2018. URL: https://www.istqb.org/documents/Turkey%20Software%20Quality%20Report%202017-2018.pdf (visited on 11/25/2021).

[Tur19] Turkish Testing Board. *Turkey Software Quality Report 2018-2019*. 2019. URL: https://www.istqb.org/documents/Turkey%20Software%20Quality%20Report%202018-2019.pdf (visited on 11/25/2021).

[Vuk+20] V. Vukovic, J. Djurkovic, M. Sakal, and L. Rakovic. "An empirical investigation of software testing methods and techniques in the province of Vojvodina". In: *Tehnički vjesnik* 27.3 (2020). Publisher: Strojarski fakultet u Slavonskom Brodu, pp. 687–696.

[WG18] D. Wang and M. Galster. "Development processes and practices in a small but growing software industry: A practitioner survey in New Zealand". In: *International Symposium on Empirical Software Engineering and Measurement*. ACM, 2018.

[ZPZ18] Q. Zhu, A. Panichella, and A. Zaidman. "A systematic literature review of how mutation testing supports quality assurance processes". In: *Software Testing Verification and Reliability* 28.6 (2018). Publisher: Wiley Online Library.

Appendix A Extracted data

This appendix only contains table A.1, which presents the all of the data extracted from the surveyed studies.

Table A.1: The data which was extracted from the studies. Values displayed in *italics* are estimates based on plotted data without explicit labeling of data points. Numerical values under findings are percentages, specifically, a percentage of respondents unless otherwise specified.

Study metadata	
Assigned identifier	1
Title	"A survey of software testing practices in Alberta"
Year	2004
Author(s)	Geras, Smith, and Miller
Reference	[GSM04]
Survey metadata	
Year(s) conducted	2002
Responses analyzed	Almost 60
Target population	Software organizations
Region/scope	Alberta, Canada
Findings	
Test level usage (management/non-management) (RQ1)	
• Unit	*73/92*
• System	*90/92*
• Accept	*70/80*
Test type usage (management/non-management) (RQ2)	
• Alpha	*40/40*
• Regression	*66/72*
• Security	*37/44*
• Stress	*63/60*
• Usability	*53/24*
Test design technique usage (RQ3)	
• Tester skill	*91*
• Customer requirements	*72*
• Out of Range	*52*
• Risks	*43*
• Boundary Values	*28*
• States	*17*
• Decision Tables	*10*
• Equivalence	*5*

- Flow Graphs *5*
- Cause Effect Graphs *3*

Defect tracking tool usage (RQ4) *68*

Automation of test levels/types (RQ4)

- Unit *32*
- Integration *11*
- System *22*
- Regression *26*
- Stress *29*

Study metadata	
Assigned identifier	2
Title	"A preliminary survey on software testing practices in Australia"
Year	2004
Author(s)	Ng et al.
Reference	[Ng+04]
Survey metadata	
Year(s) conducted	2002-2003
Responses analyzed	65
Target population	Software organizations
Region/scope	Australia
Findings	
Test type usage (RQ2)	
• UAT	47.7
• Regression testing	69.2
Test design technique usage (RQ3)	
• Black-box	44.6
• White-box	24.6
• Data-flow analysis	27.7
• Mutation analysis	4.6
• Symbolic analysis	0
Some testing activities are automated using tools (RQ4)	67.7
How those who use testing tools acquire them (RQ4)	
• Commercially	68.2
• Develop tools in-house	13.6
• Out-sourcing tool development	2.3
Tool usage (RQ4)	
• Test execution	53.8
• Regression testing	50.1
• Test analysis and reporting	41.5
• Generating test cases/scripts	30.8
• Test planning/management	26.2

Study metadata	
Assigned identifier	3
Title	"A replicated survey of software testing practices in the Canadian province of Alberta: What has changed from 2004 to 2009?"
Year	2010
Author(s)	Garousi and Varma
Reference	[GV10]
Survey metadata	
Year(s) conducted	2009
Responses analyzed	53
Target population	Software organizations
Region/scope	Alberta, Canada
Findings	
Test level usage (RQ1)	
• Unit	96
• Integration	*37*
• System	97
• Accept	*88*
Test type usage (RQ2)	
• Alpha	*50*
• Regression	*68*
• Security	*13*
• Stress	*57*
• Usability	*47*
Test design technique usage (RQ3)	
• Tester skill	*91*
• Customer requirements (user stories)	*78*
• Out of Range	*47*
• Risks	*20*
• Boundary Values	*45*
• States	*22*
• Decision Tables	*8*
• Equivalence classes	*16*
• Flow Graphs	*10*
• Cause Effect Graphs	*10*
Automation of test levels/types (RQ4)	
• Unit	*65*
• Integration	*27*
• System	*45*

- Regression *45*
- Stress *20*

Tool usage (RQ4)

- JUnit *30*
- IBM Rational test products *21*
- NUnit *21*
- FitNesse *17*
- Watin *4*
- Parasoft test products *0*
- Microsoft test products *0*
- Watir *0*

Study metadata	
Assigned identifier	4
Title	"An industrial survey on contemporary aspects of software testing"
Year	2010
Author(s)	Causevic, Sundmark, and Punnekkat
Reference	[CSP10]
Survey metadata	
Year(s) conducted	2009
Responses analyzed	83
Target population	Software companies
Region/scope	Not explicit; possibly Europe
Findings	
Test level/type usage (0 - never, 7 always)	
• Unit (RQ1)	mode 5 and 7, median 5
• Performance (including load and stress) (RQ2)	mode 7, median 5
• Functional black-box system (RQ1) (RQ2) (RQ3)	mode 4, median 6
Notes	
Results broken down by domain not extracted due to small sample	

Study metadata	
Assigned identifier	5
Title	*Software Test in Practice*
Year	2011
Author(s)	Haberl et al.
Reference	[Hab+11]
Survey metadata	
Year(s) conducted	2011
Responses analyzed	1623
Target population	Software professionals
Region/scope	Germany (77%), Switzerland (13%), Austria (10%)
Findings	
Test type usage (RQ2)	
• Regression test	80.1
• Smoke test	59.5
• Load/performance test	50
• Migration test	26
• Security/penetration test	26
Usage of black-box test design techniques (RQ3)	
• Functionality testing or function coverage	82.2
• Use case testing	77.1
• Boundary value analysis	67.6
• Equivalence partitioning	60.9
• Random testing	38.9
• State transition testing	38.3
• Decision table testing	33.1
• Pairwise testing	15.2
• Classification tree method	12.7
• Other black-box techniques	13.7
• No black-box techniques	3.5
Usage of white-box test design techniques (RQ3)	
• Statement coverage	41.9
• Decision/branch coverage	35.8
• Condition coverage	23.2
• Path coverage	22.1
• Multiple condition coverage	13.2
• Condition determination coverage	13.2
• Other white-box techniques	13.7
• No white-box techniques	3.5
Usage of experience-based test design techniques (RQ3)	

• Error guessing	84.6
• Exploratory testing without test objectives	66.1
• Exploratory testing with test objectives	33.3
• Fault attack with defect checklists	21.5
• Other experience-based test techniques	15.2
• No experience-based test techniques	5.5
Comparison of expected/actual result is automated (RQ4)	22
Automation level of test types/levels (mode, median) (RQ4)	
• Unit	70/85, 70/85
• Integration	15/30, 15/30
• System	15/30, 15/30
• Acceptance	0 15/30
Test tools are used in projects (mode median) (RQ4)	
• Developers	3/3, 2/3
• Testers	2/3, 2/3

Study metadata	
Assigned identifier	6
Title	"Survey on software testing practices"
Year	2012
Author(s)	J. Lee, Kang, and D. Lee
Reference	[LKL12]
Survey metadata	
Year(s) conducted	Unknown
Responses analyzed	24
Target population	Software professionals
Region/scope	Mostly USA
Findings	
Tools are used for activity (RQ4)	
• Test estimation	43
• Test planning	48
• Unit testing	26
• Integration testing	34
• System testing	47
• Acceptance testing	25
• Defect management	90
• Defect tracking	76
• Test reporting	76

Study metadata	
Assigned identifier	7
Title	*Turkey Software Quality Report 2011-2012*
Year	2012
Author(s)	Turkish Testing Board
Reference	[Tur12]
Survey metadata	
Year(s) conducted	2011-2012
Responses analyzed	Unknown
Target population	Software professionals
Region/scope	Turkey
Findings	
Usage of test types (RQ2)	
• Functional	98.1
• Performance	83.0
• Usability	43.4
Test design approach (RQ3)	
• Creating and running test cases	20.8
• Ad hoc testing	50.3
• Both	28.9
Utilization level of used test automation tools (mode, median) (RQ4)	Less than 50%, Less than 50%
Purpose of test automation (RQ4)	
• Bug tracking	64
• Functional/Regression testing	58
• Performance testing	44
• Test case management	40
• Requirements management	24

Study metadata	
Assigned identifier	8
Title	"A survey of software testing practices in Canada"
Year	2013
Author(s)	Garousi and Zhi
Reference	[GZ13]
Survey metadata	
Year(s) conducted	2010
Responses analyzed	246
Target population	Software practitioners
Region/scope	Canada
Findings	
Usage of test levels (RQ1)	
• Unit	80
• Functional/system (RQ2)	78
Usage of test types (RQ2)	
• Performance	*56*
• GUI	*61*
• UAT	*56*
Usage of test case generation techniques (RQ3)	
• Other	*12*
• No explicit test case generation technique	*38*
• Exploratory testing	*5*
• Code coverage (white-box testing)	*22*
• Model-based techniques (e.g., based on UML)	*15*
• Boundary Value Analysis	*28*
• Category partitioning (i.e., equivalence classing)	*24*
• Mutation testing	*29*
Average ratio of automated to manual testing (RQ4)	20/80
Usage of tools and frameworks (RQ4)	
• Other tools and frameworks	*28*
• Web application testing tools	*35*
• Family of commercial functional web tools (non-web)	*41*
• XUnit frameworks (e.g., JUnit, NUnit)	*63*
Notes	
Figure 15 differs from body text on functional/system testing. Extracted data from body text.	

Study metadata	
Assigned identifier	9
Title	*Turkey Software Quality Report 2012-2013*
Year	2013
Author(s)	Turkish Testing Board
Reference	[Tur13]
Survey metadata	
Year(s) conducted	2012-2013
Responses analyzed	Unknown
Target population	Software professionals
Region/scope	Turkey
Findings	
Test process is automated (RQ4)	
• Test management	32.5
• Test execution	36.4
• Test design	13.0
• Unit testing	33.8
• Static analysis	15.6
• Code profiling	15.6
• Performance testing and simulation	42.9

Study metadata	
Assigned identifier	10
Title	"A survey on unit testing practices and problems"
Year	2014
Author(s)	Daka and Fraser
Reference	[DF14]
Survey metadata	
Year(s) conducted	2013
Responses analyzed	171
Target population	Software developers
Region/scope	Worldwide, mostly USA
Findings	
Usage of mutation analysis (mode, median) (RQ3)	Frequently, Frequently
Usage of automated test generation (mode, median) (RQ4)	Frequently, Frequently
What is automated unit test generation used for (RQ4)	
• Exercising specifications (e.g., JML, code contracts)	33
• Exercising assertions in code	38
• Exercising parameterised unit tests	38
• Finding crashes and undeclared exceptions	41
• To complement manually written tests	8
• Other	8

Study metadata	
Assigned identifier	11
Title	*Turkey Software Quality Report 2013-2014*
Year	2014
Author(s)	Turkish Testing Board
Reference	[Tur14]
Survey metadata	
Year(s) conducted	2013-2014
Responses analyzed	Unknown
Target population	Software professionals
Region/scope	Turkey
Findings	
Usage of test techniques (RQ3)	
• Use case testing	76.2
• Checklist based	60.3
• Error guessing	54.0
• Exploratory testing	49.2
• Boundary value analysis	41.3
• Decision table	31.7
• State transition	23.8
• Attacks	22.2
• Statement coverage	20.6
• Decision coverage	20.6
• Equivalence partitioning	15.9
• Classification tree	11.1
• Pair-wise testing	9.5

Study metadata	
Assigned identifier	12
Title	"A survey of software engineering practices in Turkey"
Year	2015
Author(s)	Garousi et al.
Reference	[Gar+15a]
Survey metadata	
Year(s) conducted	2013
Responses analyzed	202
Target population	Software practitioners
Region/scope	Turkey
Findings	
Test level usage (mode, median) (RQ1)	
• Unit	Frequently, Sometimes
• Integration	Always, Frequently
• Functional/system	Always, Frequently
Test type usage (mode, median) (RQ2)	
• UAT	Always, Frequently
• Performance testing	Sometimes, Sometimes
• Load/stress testing	Seldom, Sometimes
• Security testing	Sometimes, Sometimes
Usage of test design techniques (RQ3)	
• No explicit test-case generation technique	49
• Source code analysis	33
• Model-based techniques	22
• Boundary value analysis	*24*
• Category partitioning	*24*
Automated tool usage for activities (mode, median) (RQ4)	
• Code inspections	Frequently, Sometimes
• Static code analysis	Frequently, Sometimes
Usage of test automation (mode, median) (RQ4)	Seldom, Seldom

Study metadata	
Assigned identifier	13
Title	*ISTQB® Worldwide Software Testing Practices Survey 2015-16*
Year	2016
Author(s)	ISTQB
Reference	[IST16]
Survey metadata	
Year(s) conducted	2015
Responses analyzed	3200
Target population	Software professionals
Region/scope	Worldwide
Findings	
Usage of non-functional test types (RQ2)	
• Performance	63
• Usability	56.1
• Security	38.5
• Reliability	30.7
• Accessibility	29.1
• Testability	27.7
• Efficiency	25.9
• Availability	25.6
• Maintainability	18.9
• Interoperability	18.5
• Operability	17.5
• Scalability	15
• Recoverability	12.9
• Portability	10.6
• Supportability	8.4
• Extensibility	5.2
Usage of test techniques (RQ3)	
• Use case testing	70.8
• Exploratory testing	66.3
• Checklist based	54.1
• Boundary value analysis	48.2
• Error guessing	37
• Equivalence partition	34
• Decision tables	21.5
• State transition	21.4
• Statement coverage	18.2
• Pair-Wise testing	13.2

• Attacks	10.4
• Classification tree	7.1
• Other	2.2
Usage of tools (RQ4)	
• Defect tracking	81.2
• Test execution	70.0
• Test automation	67.3
• Test management	65.4
• Performance testing	55.2
• Test design	49.7
• Requirements traceability	47.4
• Unit testing	42.9
• Static analysis	26.4
• Dynamic analysis	16.0
• Other	1.6
Percentage of test cases automated (mode, median) (RQ4)	Less than 20%, Less then 20%

Study metadata	
Assigned identifier	14
Title	*Turkey Software Quality Report 2015-2016*
Year	2016
Author(s)	Turkish Testing Board
Reference	[Tur16]
Survey metadata	
Year(s) conducted	2015-2016
Responses analyzed	Unknown
Target population	Software professionals
Region/scope	Turkey
Findings	
Usage of tools (RQ4)	
• Performance testing tools	66.4
• Monitoring tools	51.9
• Emulators / simulators	34.9
• Virtual servers	29.2
• Application performance management - APM tools	18.6
• Service virtualization tools	18.0
• Application profilers	16.9

Study metadata	
Assigned identifier	15
Title	"Toward the characterization of software testing practices in South America: looking at Brazil and Uruguay"
Year	2017
Author(s)	Dias-Neto et al.
Reference	[Dia+17]
Survey metadata	
Year(s) conducted	2015
Responses analyzed	150
Target population	Testing practitioners
Region/scope	Southern/Brazil (37%), Northern/Brazil (33%), Uruguay (30%)
Findings	
Usage of test levels (NB, SB, UY) (RQ1)	
• Unit	71, 79, 78
• Integration	79, 83, 86
• System	88, 89, 88
• Acceptance	83,82,85
Usage of test types (RQ2)	
• Regression	84, 83, 84
• Performance	70, 68, 71
• Security	70, 68, 74
Usage of exploratory testing (RQ3)	85, 77, 82
Usage of tools for activities (RQ4)	
• Test database for reuse	81, 88, 91
• Automatic execution of test procedures or cases	78, 81, 81
• Automatic generation of test procedures or cases	69, 71, 73
• Track and record results	83, 84, 88
• Estimate test effort and/or schedule	73, 75, 74
• Enact activities and artifacts	78, 80, 76
• Recording defects and the effort to fix them (bug tracking)	89, 89, 92
• Coverage measurement	71, 78, 68
• Continuous integration for automated tests	75, 82, 70
• Selection of test tools according to project characteristics	83, 78, 83

Study metadata	
Assigned identifier	16
Title	"Software Testing: The State of the Practice"
Year	2017
Author(s)	Kassab, Defranco, and Laplante
Reference	[KDL17]
Survey metadata	
Year(s) conducted	2015
Responses analyzed	167
Target population	Software professionals
Region/scope	Seemingly USA
Findings	
Usage of test levels (RQ1)	
• Unit	*72*
• Integration	*67*
• System	81
• Acceptance	*72*
Usage of regression resting (RQ2)	*63*
Usage of test types (for system tests) (RQ2)	
• Regulatory	*13*
• Documentation	*28*
• Reliability	*41*
• Scalability	*19*
• Interoperability	*21*
• Load and stability	*28*
• Stress	*30*
• Performance	*63*
• Functionality	*91*
Usage of test design techniques (RQ3)	
• Black-box	78
• White-box	42
Automatic comparison of actual and expected results (RQ4)	34

Study metadata	
Assigned identifier	17
Title	"Software quality assurance during implementation: Results of a survey in software houses from Germany, Austria and Switzerland"
Year	2017
Author(s)	Felderer and Auer
Reference	[FA17]
Survey metadata	
Year(s) conducted	2015
Responses analyzed	57
Target population	Software houses
Region/scope	Germany (33%), Austria (33%) and Switzerland (33%)
Findings	
Usage of unit testing (RQ1)	68
Usage of tools (RQ4)	
• Static code analysis	37
• Bug tracking	84

Study metadata	
Assigned identifier	18
Title	"Software testing: Survey of the industry practices"
Year	2018
Author(s)	Hynninen et al.
Reference	[Hyn+18]
Survey metadata	
Year(s) conducted	2017
Responses analyzed	33
Target population	Software organizations
Region/scope	Finland
Findings	
Testing tool usage during activities (RQ4)	
• Bug/Defect reporting	72.2
• Test automation	66.7
• Unit testing	57.6
• Bug/Code tracing	57.6
• Performance testing	48.5
• Test case management	45.5
• Integration testing	45.5
• Virtual test environment	42.5
• Automated metrics collector	36.4
• System testing	27.3
• Security testing	24.2
• Test completeness	24.2
• Test design	15.2
• Protocol/Interface conformance tool	9.1

Study metadata	
Assigned identifier	19
Title	"Development processes and practices in a small but growing software industry: A practitioner survey in New Zealand"
Year	2018
Author(s)	Wang and Galster
Reference	[WG18]
Survey metadata	
Year(s) conducted	2017-2018
Responses analyzed	101
Target population	Software developers
Region/scope	New Zealand
Findings	
Usage of test levels (RQ1)	
• Unit	73
• Integration	48
• System	26
• Acceptance	45
Usage of UI testing (RQ2)	35
Usage of automated testing (RQ4)	62

Study metadata	
Assigned identifier	20
Title	*ISTQB® Worldwide Software Testing Practices Survey 2017-18*
Year	2018
Author(s)	ISTQB
Reference	[IST18]
Survey metadata	
Year(s) conducted	2017-2018
Responses analyzed	2000
Target population	Software professionals
Region/scope	Worldwide
Findings	
Test type usage (RQ2)	
• Functional	83.0
• Performance	60.7
• Security	44.6
• Usability	44.1
• Accessibility	28.2
• Reliability	22.4
• Testability	20.5
• Availability	19.8
• Maintainability	19.3
• Efficiency	18.8
• Scalability	15.5
• Interoperability	15.4
• Operability	12.8
• Portability	11.1
• Recoverability	10.4
• Supportability	6.7
• Extensibility	4.2
• UAT	66
Test design technique usage (RQ3)	
• Use case testing	73.0
• Exploratory testing	67.2
• Boundary value analysis	52.3
• Checklist based	49.7
• Error guessing	36.0
• Equivalence partitioning	36.0
• Decision tables	28.9
• Decision coverage	25.1

• Statement coverage	21.6
• State transition	20.7
• Pair-wise Testing	13.4
• Attacks	9.3
• Classification tree	6.4
• Other	2.1
Tool usage (RQ4)	
• Defect tracking	80.6
• Test automation	70.1
• Test execution	69.7
• Test management	65.2
• Performance testing	51.2
• Requirements traceability	45.8
• Test design	43.9
• Unit testing	43.8
• Static analysis	27.9
• Dynamic analysis	17.2
• Other	2
Percentage of test cases automated (mode, median) (RQ4)	1-10%, 11-20%

Study metadata	
Assigned identifier	21
Title	*Turkey Software Quality Report 2017-2018*
Year	2018
Author(s)	Turkish Testing Board
Reference	[Tur18]
Survey metadata	
Year(s) conducted	2017-2018
Responses analyzed	300
Target population	Software professionals
Region/scope	Turkey
Findings	
Usage of test automation for test levels/types (RQ4)	
• Regression	35
• UI	25
• Integration	18
• Unit	18
Tool usage (RQ4)	
• Test management	70
• Test automation	57
• Defect tracking	52
• Test data management	24
• CI/DevOps tools	23
• Static analysis	18
• Dynamic analysis	16
• Test virtualization	9
• Other	1

Study metadata	
Assigned identifier	22
Title	"A survey of software testing practices in Costa Rica"
Year	2019
Author(s)	Quesada-López, Hernandez-Agüero, and Jenkins
Reference	[QHJ19]
Survey metadata	
Year(s) conducted	2018
Responses analyzed	92
Target population	Software professionals
Region/scope	Costa Rica
Findings	
Usage of test levels (RQ1)	
• Unit	79
• Integration	81
• System	83
• Acceptance	88
Usage of test types (RQ2)	
• Performance	71
• Security	71
• Regression	79
Usage of exploratory testing (RQ3)	69
Usage of tools for activities (RQ4)	
• Test database for reuse	77
• Automatic execution of test procedures or cases	71
• Automatic generation of test procedures or cases	58
• Track and record results	74
• Estimate test effort and/or schedule	62
• Enact activities and artifacts	73
• Recording defects and the effort to fix them (bug tracking)	82
• Coverage measurement	61
• Continuous integration for automated tests	66
• Selection of test tools according to project characteristics	67

Study metadata	
Assigned identifier	23
Title	*Turkey Software Quality Report 2018-2019*
Year	2019
Author(s)	Turkish Testing Board
Reference	[Tur19]
Survey metadata	
Year(s) conducted	2018-2019
Responses analyzed	Unknown
Target population	Software professionals
Region/scope	Turkey
Findings	
Test type usage (RQ2)	
• Functionality	92
• Performance	64
• Usability	62
• Accessibility	35
• Testability	34
• Security	33
• Availability	28
• Reliability	26
• Maintainability	25
• Operability	13
• Portability	13
• Interoperability	12
• Recoverability	11
• Supportability	8
• Extensibility	8
• Scalability	8
Test technique usage (RQ3)	
• Use case testing	69
• User story testing	61
• Checklist based	61
• Error guessing	43
• Exploratory testing	42
• Boundary value analysis	38
• Decision table	26
• Equivalence partitioning	25
• Decision coverage	22
• State transition	19
• Statement coverage	16

• Classification tree	15
• Attacks	13
• Pair-wise testing	9
• Other	1
Tool usage for activities (RQ4)	
• Defect tracking	59
• Test management	58
• Performance testing	50
• Test execution	49
• Unit testing	38
• Test design	37
• Requirements traceability	31
• Static analysis	24
• Dynamic analysis	11
• Other	2
Degree of test automation (mode, median) (RQ4)	1-10%, 11-20%

Study metadata	
Assigned identifier	24
Title	"A preliminary survey on software testing practices in Khyber PakhtunKhwa region of Pakistan"
Year	2020
Author(s)	Latif and Rana
Reference	[LR20]
Survey metadata	
Year(s) conducted	2018
Responses analyzed	70
Target population	Software organizations
Region/scope	Khyber PakhtunKhwa, Pakawistan
Findings	
Most used box approach (RQ3)	
• White-box	57.1
• Black-box	*32.9*
• Grey-box	*11.4*
Most used test design technique (RQ3)	
• Tester skills	58.6
• Boundary value analysis	8.6
• Equivalence partitioning	11.4
• Control flow graph	21.4
Notes	
The questionnaire in this survey is designed differently than the other ones in that for example, rather than asking respondents which test design techniques they used, respondents were asked to select their most used test design technique (single choice). Using this approach requires that the options are mutually exclusive and comprehensive; data about test levels and test tools were not extracted due to problems in this regard.	

Study metadata	
Assigned identifier	25
Title	"An empirical investigation of software testing methods and techniques in the province of Vojvodina"
Year	2020
Author(s)	Vukovic et al.
Reference	[Vuk+20]
Survey metadata	
Year(s) conducted	Unknown
Responses analyzed	83
Target population	Software professionals
Region/scope	Vojvodina, Serbia
Findings	
Usage of test design techniques (RQ3)	
• Black-box	92.2
• White-box	78.4
• Equivalence class partitioning	21.7
• Boundary value analysis	60.2
• Combinatorial test techniques	49.4
• Cause-Effect graphing	24.1
• Statement coverage	48.2
• Branch Coverage	53.0
• Condition Coverage	61.4
• Loop Coverage	55.4
• Control Flow Graphs	44.6
• Code Complexity Analysis	36.1
• Exploratory testing (mode, median)	Medium agreement, Medium agreement

Study metadata	
Assigned identifier	26
Title	"Experiences and Practices in GUI Functional Testing: A Software Practitioners' View"
Year	2021
Author(s)	Junior et al.
Reference	[Jun+21]
Survey metadata	
Year(s) conducted	2021
Responses analyzed	222
Target population	Testing professionals
Region/scope	Brazil
Findings	
Tools/frameworks used for GUI testing (RQ4)	
• Selenium	35.97
• Appium	20.95
• Cypress	9.49
• Others	7.51
• TestComplete	5.14
• Sikuli	3.95
• SilkTest	3.16
• Robot Framework	2.77
• Microsoft Coded UI Tests	2.37
• Oracle Application Testing Suite	1.58
• UFT	1.58
• IBM Rational Functional Tester	1.19
• Ranorex	1.19
• Protractor	0.79
• TestCafe	0.79
• VS Code	0.79
• Webdriver IO	0.79
Usage of automation in GUI tests (RQ4)	
• Only automation	9
• Some automation, some manual	41
• No automation	52
Automation of activities (RQ4)	
• API tests	37
• Integration tests	28
• Performance tests	21
• Unit tests	12

- Regression tests 0.5
- Chatbot tests 0.5
- Security 0.5

Tools used for reporting bugs found using manual tests (RQ4)

- Kanban board 48.2
- Jira 21.6
- Slack message 13.1
- Others 7.2
- Mantis 7.2
- Spreadsheets 5.9
- Azure DevOps 4.5
- E-Mail 4.5
- AML 2.3
- Microsoft Teams 0.9
- Silk 0.9
- Trello 0.9
- Zephyr 0.9
- Corporate platform 1.4

Tools used for reporting bugs found using automated tests (RQ4)

- Kanban board 27.0
- Slack message 10.4
- Jira 8.6
- Others 7.7
- E-mail 5.9
- Mantis 3.2
- Azure DevOps 2.3
- Direct Messenger 1.4
- Redmine 0.9
- Reports 0.9

Study metadata	
Assigned identifier	27
Title	"Practices in software testing in Cameroon challenges and perspectives"
Year	2021
Author(s)	Maxime Carlos and Ibrahim
Reference	[MI21]
Survey metadata	
Year(s) conducted	Unknown
Responses analyzed	30
Target population	Software companies
Region/scope	Cameroon
Findings	
Usage of test levels (RQ1)	
• Unit	73.33
• Integration	76.67
• System	26.67
• Acceptance	26.67
Usage of test types (RQ2)	
• Ramp-up/scalability test	3.33
• Recovery test	6.67
• Compatibility test	13.33
• Conformity test	26.67
• Security test	43.33
• Performance test	43.33
• Regression testing	53.33
• Usability test	63.33
Techniques for identifying test scenarios (RQ3)	
• Technique based model	10
• White box test	10
• Partitioned analysis	13.33
• Value limit analysis	30.00
• Exploratory test	30.00
• No explicit technique	60.00
Frequency of mutation test usage (mode, median) (RQ3)	Never, Never
Development of test tools for activities (RQ4)	
• None	83.33
• Test execution	6.66
• Regression testing	6.66
• Analysis and production of results	3.33
• Test data generation	3.33

- Scenarios generation/Scripts test 0
- Planning/tests management 0

Usage automatic testing tools (RQ4) 40

Usage of specific tools (RQ4)

- Selenium 40
- Sonarqube 15
- FindBugs 10
- XStubio 5
- JUNIT 5
- JENNY 5
- JMeter 5
- HP ALM 0
- Testlink 0
- HP UFT 0
- NeoLoad 0
- SlideShare 0