What 5 Million Job Advertisements Tell Us about Testing: a Preliminary Empirical Investigation

Maura Cerioli, Maurizio Leotta, Filippo Ricca

Abstract:

Software testing is a crucial part of business success to ensure final product quality. However, little concrete data exists on technical demands about it in the industry, mostly collected through personal opinion surveys on a restricted sample of professionals.

In this paper, we used a different approach: we applied content analysis to a set of about five million job advertisements taken from a popular Web job-search engine. The analysis of job advertisements is more promising than surveys because the data are by far more numerous and distributed geographically.

The content analysis results revealed four essential findings on the current practice of software testing: a) Companies search for about six times more Coders than Testers, b) Unit testing is the most required skill for Coders while Acceptance testing is the most popular for Testers, c) Automated testing dominates the job advertisement scene compared to Manual testing and, d) the most valuable testing tools and frameworks are Selenium, JUnit, and Cucumber for both Testers and Coders. We believe that these findings (and other related results from the content analysis study) will be useful for professionals, instructors, and researchers dealing with software testing.

Digital Object Identifier (DOI):

https://doi.org/10.1145/3341105.3373961

https://doi.org/10.1145/3341105.3373961

What 5 Million Job Advertisements Tell Us about Testing: a Preliminary Empirical Investigation

Maura Cerioli
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università di Genova, Italy
maura.cerioli@unige.it

Maurizio Leotta
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università di Genova, Italy
maurizio.leotta@unige.it

Filippo Ricca
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università di Genova, Italy
filippo.ricca@unige.it

ABSTRACT

Software testing is a crucial part of business success to ensure final product quality. However, little concrete data exists on technical demands about it in the industry, mostly collected through personal opinion surveys on a restricted sample of professionals.

In this paper, we used a different approach: we applied content analysis to a set of about five million job advertisements taken from a popular Web job-search engine. The analysis of job advertisements is more promising than surveys because the data are by far more numerous and distributed geographically.

The content analysis results revealed four essential findings on the current practice of software testing: a) Companies search for about six times more Coders than Testers, b) Unit testing is the most required skill for Coders while Acceptance testing is the most popular for Testers, c) Automated testing dominates the job advertisement scene compared to Manual testing and, d) the most valuable testing tools and frameworks are Selenium, JUnit, and Cucumber for both Testers and Coders. We believe that these findings (and other related results from the content analysis study) will be useful for professionals, instructors, and researchers dealing with software testing.

CCS CONCEPTS

• **Software and its engineering → Empirical software validation**; **Software testing and debugging**.

KEYWORDS

Software Testing, Content analysis, Tools and Frameworks, Job Advertisements, Industrial Practice.

ACM Reference Format:
Maura Cerioli, Maurizio Leotta, and Filippo Ricca. 2020. What 5 Million Job Advertisements Tell Us about Testing: a Preliminary Empirical Investigation. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3341105.3373961

SAC '20, March 30-April 3, 2020, Brno, Czech Republic

ACM ISBN 978-1-4503-6866-7/20/03…$15.00
https://doi.org/10.1145/3341105.3373961

1 INTRODUCTION

Software testing is an essential and complex technical activity that requires skilled professionals proficient in many different methodologies, tools, practices, and techniques performing diverse roles. Inadequate software testing usually leads to significant risks and serious consequences (e.g. failures as the Mars Polar Lander, the Patriot missile, and the Therac 25 radiation deaths [13]). For this reason, it is vital to know the state of the art and practice and learn from it. We must study the best practices, procedures, methodologies, and tools used in industrial practice, software-testing problematic areas, possible solutions, and new trends. Knowing the current practice to improve the future is paramount for managers that have to decide which methodologies, tools, and techniques are the most suitable. Also, instructors, researchers, and academics dealing with the long-standing problem of teaching useful and current software testing topics, or interested in finding challenging research topics need the same practical knowledge.

Unfortunately, little concrete data exists on software-testing and quality assurance practices [8]. Moreover, most of them were collected using surveys [15] on restricted samples of professionals. Researchers use surveys to gather relevant information from a sample of people, aiming at generalizing the results to a larger population. However, small samples, with respondents often selected from a few countries, could be non-representative and make generalizations difficult, affecting the study's external validity [14].

In this paper, we use a different strategy, that is, content analysis (see e.g. [21]) applied to a massive set of job advertisements taken from a popular Web job-search engine. The method of analyzing job adverts is promising because the data are numerous, distributed geographically, easily accessible, and well represent the industry needs. In our case, we analyzed about five million job advertisements published from 2015 up to 2018, written in English, and located in 220 different countries.

The long term goal of our work is taking a snapshot of the industrial practice in the field of software testing to answer research questions of interest for teachers, researchers, and professionals. This paper is just the first step of this quest, where we investigate: (a) the relevance of the testing phase in the software development, e.g., when compared to coding; (b) which testing categories are the most used in practice, e.g., among unit, integration, and system testing; (c) the prevalence of manual testing vs. automated testing; and finally (d) which are the frameworks and tools most adopted in the industry. This last point is of high relevance because people are the best asset in the IT industry, and giving them the right tools will boost their productivity and provide the business an edge.

The paper is organized as follows: Section 2 describes our empirical study, presenting goals, research questions, and data set. Adopted analysis procedure and preliminary results are shown in Section 3 and 4, respectively. Finally, related works are discussed in Section 5, and we draw our conclusions in Section 6.

2 STUDY DEFINITION

The main aim of this work is collecting information about industrial practice on software testing.

2.1 Method

Usually, researchers answer this kind of question by mean of personal opinion surveys that need many respondents to be significant (page 12, [22]). Indeed, a limited number of participants reduces the reliability of the obtained results from a statistical viewpoint. Thus, one of the main limitations of this approach lies in the difficulty of reaching enough interviewees. For instance, in our previous experience [11, 16, 20], we found getting more than a few hundred participants very difficult. Analogously, other researchers in the software engineering field experienced the same difficulties. For instance, [3] is based on 178 professionals, [2] on 97, and [12] on 48.

Therefore, in this study, we decided to face the problem differently. Nowadays, job search engines on the web collect a massive number of job advertisements (ads). Each ad can contain a detailed description of the job and the required skills and technologies. Our idea is to use such a bounty of data to gather insight into several aspects of the testing activities in the current industrial practice.

We made an agreement with LinkUp[1], one of the most extensive indexes of job openings sourced directly from company websites globally (more than 100 million of jobs indexed in the last decade, and 3.5 million daily active job count), getting access to their data for research purposes.

2.2 Goal

The ***goal*** of this work is *taking a snapshot of the industrial practice in the field of software testing* to the interest and use of:

- *teachers and instructors* interested in offering courses and tutorials on the techniques, technologies, and tools most requested and used in industrial practice;
- *researchers* interested in focusing their research activities and efforts on the testing aspects most relevant for the industry;
- *SQA managers and testers* interested in understanding what are the testing techniques, frameworks, and tools most relevant in the industry, in particular when they have to make critical decisions about introducing something new in their organizations;

2.3 Research Questions

Starting from the above goal, we derived the following *Research Questions* (RQs):

RQ1: *Which is the relevance of software testing compared to the relevance of coding?*

By comparing testing with coding, the first RQ aims at understanding the importance of testing in the context of software development. For answering this RQ, we used as a proxy the ratio between the number of job ads (contained in the considered dataset) requiring software testers and coders, respectively.

The ratio of testers to coders, which represents how many times the industry sought software testers compared to coders, let us infer the market share of software testers in contrast to coders. From it, we can get a rough idea of the relevance of software testing compared to coding.

RQ2: *Which categories/types of software testing are the most required among Coders and Testers?*

The second RQ aims at assessing which categories/types of testing, such as unit, acceptance, End-to-End (E2E), performance/load, security/penetration, are the most required, and so perceived as more useful, from the companies. For answering this RQ, we used as a proxy the percentage of job offers for Coders and Testers requiring specific categories/types compared to their total. Section 3 describes the procedure to define the categories of testing considered in this study.

RQ3: *Which of the two categories, manual and automated testing, is nowadays the most required among Coders and Testers?*

Testing may be classified into two broad categories: manual and automated. Manual testing is the process of checking a software product manually against functional and non-functional requirements without executing scripted tests. On the contrary, automated testing relies on the use of dedicated software tools. The third RQ aims at assessing what is the relevance of automated testing in the industrial practice, nowadays. For answering this RQ, we used as a proxy the ratio between the number of job advertisements of Coders and Testers, concerning automated and manual testing.

RQ4: *Which testing tools and frameworks are the most requested for Coders and Testers?*

The fourth RQ aims at evaluating which tools and frameworks are the most valuable in the industrial practice for testing purposes. We postulate that the number of times a job ad in a dataset requires a tool correlates with its perceived relevance. For answering this RQ, we used as a proxy the percentage of job offers of Coders and Testers requiring a specific tool/framework compared to their total. Section 3 describes the procedure to define the list of testing tools and frameworks considered in this empirical study.

2.4 Dataset

The context consists of the job advertisements available in the LinkUp dataset. Our data source is the complete dataset containing all the job advertisements published in the last decade, listing more than 100 million records from 220 different countries. The amount of data is enormous (more than 275GB overall). For each job advertisement, several useful information is included, like company name, city, state, country, creation and deletion date, full-text description, *O*NET* code.

The Occupational Information Network[2] (*O*NET*) is a free online database that contains hundreds of occupational definitions. It characterizes jobs in terms of the skills and knowledge required, how

[1] https://www.linkup.com/

[2] https://en.wikipedia.org/wiki/Occupational_Information_Network

the work is performed, and typical work settings. It was developed under the sponsorship of the US Department of Labor/Employment and Training Administration (USDOL/ETA).

We used the *O*NET* associated with each job ad to filter the dataset and drastically reduce the number of entries. Indeed, we kept only those related to the IT sector, as described in the next session.

3 DATA PREPARATION PROCEDURE

We received the data in raw textual form and extracted from the vast mass of job advertisements only those of interest. Then, we used them to populate a database specifically designed to support the queries needed for our research questions.

In Section 3.1, we will discuss how we have built the table with the data about the job advertisements, by subsequent refinements. In Section 3.2, we will present the tagging procedure adopted for associating job advertisements to keywords about testing and coding.

3.1 Dataset Cleaning

The data made available by LinkUp concern about 110 million advertisements published from 2007 up to 2018, mostly written in English, but located in 220 different countries.

The information is split into two parts. The first is the *job records* that is the data needed for a brief synopsis, like, for instance, title, seeking company, job location. The second is the *job description* that is a full text of an average length of 3,500 chars, detailing the job. Both parts use the MD5 hash of the *URL* of the job offer as the primary key. The hash also acts as the external key from *job description* to *job records*. The overall job record list occupies about 25 GB in *CSV* format, while the corresponding descriptions occupy about 250 GB in XML format.

Given the massive amount of data, we needed to filter from the very beginning the irrelevant data, to avoid wasting space and time, and make a query-intensive approach feasible.

We based the first data pruning on the categorization made by the companies. Indeed, each job record contains the *O*NET Occupation Code*, as described in the previous section. We selected only the codes having testing among the typical tasks listed on the *O*NET* website. The codes chosen in this way are the 27 in the macro-category *Computer and Mathematical* related to IT jobs (codes starting by 15-11) plus Computer and Information Systems Managers (code 11-3021.00), Quality Control Analysts (code 19-4099.01) and Computer Operators (code 43-9011.00). We deemed many selected categories irrelevant to answer our RQ accordingly to their description. However, we chose to be conservative and keep all of them, on the off chance that the presence of test in the task list misled companies seeking testers into using some of them.

Therefore, we imported into the table Jobs all the job records having any of the selected *O*NET*, and we got 11,086,286 rows (about 10% of the overall content of the original data repository). Then, we imported into the table JobDescriptions only the descriptions with the same hash key as some entry in Jobs, and we got 5,239,455 entries. Thus, less than half of the job records have a corresponding full-text long description.

From the raw imported data, we extracted a third table, Infoes, where we joined job descriptions and the relevant part of job information, to simplify and optimize the queries. We kept the columns Hash, Title, Country, Created, and Onet from Jobs, and the column Description from JobDescriptions. We added the column DescriptionLanguage, initialized using the library for language detection by Pēteris Ņikiforovs[3]. Such a library uses naive Bayesian filters to detect the language of a text and claims to have 99% over precision for 53 languages. We experimented with language detection on titles also but found it unreliable, as, in many cases, the text shortness and the massive presence of acronyms and company names prevented the library from working.

To be able to analyze the text, we needed to restrict ourselves to just one language, so we removed the 163,701 advertisements with non-English description language.

Finally, we restricted our analysis to the years 2015-2018 for two reasons. On the one hand, we are more interested in recent history, having a more significant impact on the current trends. On the other hand, technology is changing fast, so that any research question about tools needs to target a small interval.

The final count is 4,824,591 rows in Infoes. Those job ads are the data we work on in this paper. Moreover, 88% of the job records for those years have a description, hence appears in Infoes. Therefore, we work on high quality, extremely representative data set of about five million points.

3.2 Data Classification

The next step was the definition of a list of *conceptual categories* representing topics in the field of testing and coding. Each category clusters one or more keywords related to the same specific concept, be it an abstract topic or a tool. For instance, we selected the keywords *"E2E test", "end-to-end test", "end2end test"* for the conceptual category of *end to end testing*. Then, that list is expanded in our searches with variants for the word test, like, for instance, tests, tested, and testing, as we will discuss later.

We aimed to have a comprehensive list to be used to reliably classify the job advertisements regarding the subjects of our research questions. To select the categories, we integrated our experience with an analysis of a few authoritative sources, like classic textbooks and *Wikipedia*. Then, we added hits from many google searches, mostly for the tool part, to be sure to cover the current practices.

In particular, on the coding side, we examined lists of *programming tools* from *Wikipedia*, the developer surveys by StackOverflow for the years of interest[4], and the TIOBE Programming Community index[5], that sorts programming languages according to their popularity. Moreover, for each of the top 20 languages in that index, we searched for the best tools for that language and inspected the hit first page.

Similarly, on the testing side, we used the typology of testing listed on *Wikipedia*, and the tools cited in its category for *software testing tools*. Moreover, we searched for *best software testing tools* in general and specifically for the different kinds of software testing. Then, we examined the hit first page. In particular, for the

[3] https://github.com/pdonald/language-detection
[4] https://insights.stackoverflow.com/survey/201x, for x=5,6,7,8
[5] https://www.tiobe.com/tiobe-index/

automation testing tools, we are in debt with the web site Test Guild[6] by Joe Colantonio, where tools are empirically rated based on questionnaires administered to professionals.

The outcome of this investigation was a list of slightly more than 400 conceptual categories, memorized into the table Categories, and fairly divided between testing and coding. Then, we wanted to relate conceptual categories with job advertisements referring to them. Moreover, we needed to do this automatically, given the size of the population (about 5 million). So, for each conceptual category, we collected synonyms, acronyms, and linguistic variants, and used them to query the description column of the table Infoes for matches. We used the found correspondences to define a relation between Categories and Infoes. This approach entails executing millions and millions of textual matches. Thus, simply using LIKE is unfeasible because the required time would be far too much. Therefore, we used a *Full Text Indexing* (FT-index in the following) of the columns containing advertisement titles and descriptions. FT-index is a standard feature of MS SQL-Server that builds an index associating each word to the position(s) where it appears in each cell. Roughly speaking, building a *Full Text Indexing* means that for each row in the table (in our case Infoes) and each column to be indexed (in our case Title and Description)

(1) the content of that cell is tokenized in words, breaking the strings where spaces, tabs, end of lines, punctuation symbols, numbers and the such occur;
(2) the *stop words*, that is, those words that are too short or too common in a language to be significant, like for instance articles or auxiliary verbs, are dropped;
(3) for each remaining word, the index is updated, adding the reference to the row, the column, and the positions of all its occurrences.

Though this process is expensive, it is performed just once and makes subsequent queries very fast. Using an FT-index, different kinds of queries are possible:

- looking if a word or a phrase is [not] contained in the text;
- looking for a group of words all appearing in the text within a given distance, preserving or not their order; thus, we can look for a word *near* some others, tailoring the concept of *near* on a case-by-case base;
- looking for all *inflections* of words the language dictionary (in our case in the English dictionary). That is, both singular and plural for nouns, all verbal forms for verbs;
- a logical combination of the above, by the usual Boolean connectives and, or, not.

We took full advantage of all these possibilities to deal with problematic keywords.

Indeed, some conceptual categories were trivial to manage. For example, the tools with phrasal names and meaningless acronyms just generated a disjunction (or) of contains clause. For instance, Micro Focus Unified Functional Testing (UFT) software, formerly known as QuickTest Professional (QTP), generates:

```
CONTAINS( Description, '"Unified Functional Testing"
  or UFT or "HP QuickTest Professional"
  or QTP or "QuickTest Pro"' )}
```

[6]https://testguild.com/automation-testing-tools/

However, many categories were more challenging and very few unmanageable. The first cause of troubles was the possible clash between the names (or acronyms) of tools and common English nouns, like, for instance, Selenium, or Cucumber, or Espresso. Indeed, we had to distinguish name occurrences referring to the tool from those using the name in a generic sense. Moreover, some nouns were also denoting other tools by the same name in a different area, like Oxygen, or place names, like, for instance, Buffalo. In these cases, we refined the query. The first step was verifying if the clash was present in our data by manually inspecting 30 random matches for the keyword. If we did not find any false positive in the sample, we just used the standard query. Otherwise, we tried to find a difference in usage patterns between the correct cases and the false positives, in terms of other words [not] occurring in the vicinity of the keyword, and refined the query using the *near* form and those words. Then, we verified the correctness of this new candidate query by checking 30 random matches of it for false positive. Moreover, we verified that the new query was not too restrictive taking 30 random matches of the keyword not selected by the candidate query for false negative. If we did not find any in both cases, we used the candidate query. Otherwise, we refined the query differently and repeated the process until success. In very few cases, we were not able to complete the process and had to give up on the keyword, like, for instance, Buffalo[7], or Brackets[8].

The choice of 30 values for the samples was a compromise between the effort required by the manual inspection and the confidence level we wanted to achieve. Indeed, we experimented with samples of different sizes and found out that with smaller numbers, repeated samplings could present errors, even if the first selection did not. However, with a size of about 25-30 elements, the problem disappeared.

To give the gist of how refined queries look, in the case of the Espresso Testing Framework for Android by Google, we landed, after some trial and error, on the following query

```
CONTAINS(Description, '"espresso framework" or
  near((espresso,test),3,false) or
  near((espresso,framework),3,false) or
  (espresso and
   (android or Robolectric or Roboelectric or selenium)
    and not coffee and not tea and not gourmet and not
    Formsof(inflectional,"espresso machine"))')
```

The final part of the query prevents the many job advertisements explicitly mentioning free espresso and other gourmet beverages from matching. The first part captures ads relative to automated testing or Android development.

A second, totally different, kind of problem were keywords that are also *stop words*, as they are ignored both in the search phrase and in the searched text. The supreme example is the C language. Indeed C, having just one letter, is a stop word and, as such, is not even indexed. However, C as a language has great relevance when tagging job advertisements for software developers. Thus, in this and similar cases, we had no other choice but using the

[7]https://gobuffalo.io/
[8]http://brackets.io/

like construct in the query. For instance, in the case of the C language, we built the following search predicate (used in SQL SELECT statements):

```
'.'+Description+'.' like
    '%[^a-zA-Z0-9&£]C[^a-zA-Z0-9+#£]%';
```

4 RESULTS

This section reports for each RQ, the procedure followed to define the answer and the obtained results.

4.1 RQ1: Testing vs. Development

Procedure. To answer **RQ1**, we need to estimate the number of tester/coder positions offered. For this purpose, we will count the ads having *O*NET* codes representing tester or coders, respectively. An analysis of the job descriptions in the *O*NET* catalog shows that software testers belong in code *15-1199.01 - Software Quality Assurance Engineers and Testers*, and there are no other codes for them. On the other hand, a few codes characterize specific types of coders, and some programming activity is required by many less focused job categories too. Since we are interested in *full-time application coders*, the codes of our interest are *15-1132.00 - Software Developers, Applications* and *15-1134.00 - Web Developers*. A couple of other *O*NET* titles sound like a synonym for coders, but we discarded them after diving deeper in their descriptions. The first one is *15-1133.00 - Software Developers, Systems Software*, that concerns low-level programming, while we are interested in the application-level. Another one is *15-1131.00 - Computer Programmers*, which defines sort of *assistants* to software developers, with some coding responsibilities, but restricted more to scripting applications than developing them.

Results. Table 1 summarizes the most important information for answering **RQ1**. The table reports the data for the top-20 most used *O*NET* in the IT sector. The first two columns contain the *O*NET* code and the corresponding description. The third column reports the number of advertisements labeled with that *O*NET* in the years 2015-2018 (in decreasing order). We marked by **C** the *O*NET* representing coders and by **T** the one for testers. Finally, the last columns contain data to check that the *O*NET* labeling of job ads agrees with their contents, as discussed in the following subsection *Data Check*.

As we can see from the table, the *O*NET 15-1132.00 - Software Developers, Applications* is the one that scores more job advertisements in the entire IT field, with 1,126,893 advertisements. The other *O*NET* related to coders, *15-1134.00 - Web Developers*, has been used for labeling 205,507 advertisements. Thus, the total number of advertisements for coders is 1,332,400. On the other hand, the sole *O*NET* directly related to testers, *15-1199.01 Software Quality Assurance Engineers and Testers*, scores 230,676 job advertisements. Thus, to answer RQ1, we compute the ratio between the number of ads included in the coders and testers categories obtaining a value of 5.78. If we broaden our interpretation of coder to include also *15-1133.00 Software Developers, Systems Software* and *15-1131.00 Computer Programmers*, the total number of coders related advertisement raises to 1,580,982 and the ratio to 6.85.

Data Check. We cannot rule out *a priori* that job advertisements are incorrectly labeled. Indeed, the structure of the *O*NET* classification is quite complex. Thus, job advertisers could misunderstand *O*NET* definitions. To check that companies did not make mistakes (in statistically significant measure), we verified that the keyword distribution agreed with our analysis of *O*NET* classification. Thus, we computed the percentage of rows from the Infoes table that matched at least two conceptual categories related to coding/testing for each *O*NET* code. We asked for two matches (instead of one), to limit the effects of false positives. Thus, the matching advertisements have a very high probability of being related to coder/tester jobs. The results of our verification are reported in Table 1, columns *% of match* for coders and testers.

Concerning *Coders*, we can see that the highest percentages are by far those for the *O*NET* we identified as coders (labeled by a **C**), with values around 80%. The second best group consists of (i) *Software Developers, Systems Software*, (ii) *Computer and Information Research Scientists*, and (iii) *Computer Programmers*, with values around 55-60%, followed by *Software Quality Assurance Engineers and Testers*, with 41%. The others have values smaller than 25%.

				Coders Check		Testers Check	
	O*NET Code	Description	Advertisements	% of match	median	% of match	median
C	15-1132.00	Software Developers, Applications	1,126,893	77.6%	4	21.3%	0
	15-1151.00	Computer User Support Specialists	546,289	7.4%	0	1.7%	0
	15-1142.00	Network and Computer Systems Administrators	527,351	23.4%	0	3.7%	0
	15-1121.00	Computer Systems Analysts	447,594	25.4%	0	21.2%	0
	15-1199.09	Information Technology Project Managers	408,300	7.7%	0	6.6%	0
	15-1122.00	Information Security Analysts	303,179	14.7%	0	7.9%	0
	15-1199.02	Computer Systems Engineers/Architects	291,596	22.6%	0	7.0%	0
T	15-1199.01	Software Quality Assurance Engineers and Testers	230,676	41.0%	1	78.8%	5
C	15-1134.00	Web Developers	205,507	83.1%	5	15.5%	0
	11-3021.00	Computer and Information Systems Managers	214,529	6.5%	0	1.6%	0
	15-1133.00	Software Developers, Systems Software	181,079	61.6%	2	18.1%	0
	15-1131.00	Computer Programmers	67,503	54.6%	2	10.9%	0
	15-1111.00	Computer and Information Research Scientists	70,686	58.9%	2	1.2%	0
	15-1141.00	Database Administrators	60,290	24.3%	0	4.1%	0
	15-1143.00	Computer Network Architects	43,041	8.8%	0	4.5%	0
	15-1121.01	Informatics Nurse Specialists	28,052	1.7%	0	4.0%	0
	15-1199.08	Business Intelligence Analysts	18,314	14.7%	0	2.6%	0
	19-4099.01	Quality Control Analysts	15,488	0.7%	0	2.5%	0
	15-1199.10	Search Marketing Strategists	10,519	23.0%	0	1.4%	0
	43-9011.00	Computer Operators	7,507	2.6%	0	0.3%	0

Table 1: RQ1. Top-20 most used *O*NET* in the IT sector in the years 2015-2018.

The second best group percentages may seem significant. However, taking into account also the median values, we can easily see that the pertinence of the corresponding job categories is much smaller. In particular, their median value is two against the 4 or 5 for the *O*NET* we have selected.

We blame the dispersion of matches for coding concepts on the conceptual categories representing technologies. Indeed, they can also appear in advertisements for end-users/testers/support people of that technology and not only developers. Scripting languages and environments, together with activities like debugging, are the most popular matches for the excluded *O*NET* scoring high values.

The same analysis for *Testers* yields even more drastic results, as *Software Quality Assurance Engineers and Testers* reaches almost 80%, and it is the only one with values over 25%. Accordingly, its median is 5, while all the others have 0.

Both for *Coders* and *Testers*, we manually inspected 30 randomly chosen advertisements and verified that there were no false positive.

Therefore we can confirm our initial analysis about the *O*NET* representing coder and tester categories and use the number of advertisements labeled by such *O*NET* codes as an estimate of the number of coders and testers.

Summary. In conclusion, it emerges that companies search about six times more for coders than for testers.

4.2 RQ2: Testing Categories

Procedure. To answer **RQ2**, we compared the testing levels and the testing types listed on *Wikipedia*, with two modifications: (1) we added *End-to-end testing*, that is gaining nowadays more and more relevance; (2) we dropped *Development testing*, as it was more related to best development practices than to testing *per se*. *Development testing* is also too difficult to search for, getting mostly false positive. For instance it matches the phrases *development of testing cases*, *of* being a stop word, and *system/software... development. Testing...*.

Since all these testing levels and types were conceptual categories, we counted the numbers of distinct links between them and job advertisements, grouped by *O*NET*. We did not take into account the tools, as many of them serve different testing purposes. For instance, many unit testing tools are also used to run integration/acceptance tests. Moreover, integrated frameworks offering support for many different kinds of testing are quite common. Thus, it is not possible to automatically understand, from a textual match, if the advertising companies are interested in the framework for some task or the other.

Results. Table 2 summarizes our findings partitioned into two macro-categories: (1) testing levels and (2) testing types, techniques, and tactics. For each entry of the table, we report the total number of matches found in (a) all the advertisements related to Coders (i.e., 15-1132.00 and 15-1134.00) and (b) all the advertisements related to Testers (i.e., 15-1199.01). Also, we report the percentage of the Coders and Tester advertisements matching each specific entry. We list the entries of the two macro-categories in descending order of the number of matches in the column "Testers".

Concerning the "Testing Levels" (rows 1-4) and focusing on Coders, the most requested is unit testing followed by integration, acceptance, and system (see the blue bars). On the contrary, in the case of Testers, the ranking is almost reversed. Acceptance

		Number of Matches on Coders		Number of Matches on Testers	
Testing levels	Acceptance testing	31,876	2.8%	27,600	15.2%
	System testing	26,494	2.3%	21,780	12.0%
	Integration testing	57,255	4.9%	19,010	10.5%
	Unit testing	175,962	15.2%	11,912	6.6%
Testing types, techniques and tactics	Regression testing	18,260	1.6%	35,095	19.3%
	Functional testing	15,991	1.4%	25,847	14.2%
	Software performance testing	22,035	1.9%	25,102	13.8%
	End to end testing	5,895	0.5%	9,500	5.2%
	Security testing	7,503	0.6%	4,681	2.6%
	Non-functional testing	1,499	0.1%	4,295	2.4%
	Smoke and sanity testing	943	0.1%	2,885	1.6%
	Usability testing	8,072	0.7%	1,670	0.9%
	Compatibility testing	399	0.0%	1,618	0.9%
	Continuous testing	1,758	0.2%	1,382	0.8%
	Beta testing	964	0.1%	769	0.4%
	Installation testing	572	0.0%	637	0.4%
	Accessibility testing	505	0.0%	623	0.3%
	Internationalization and localization	109	0.0%	467	0.3%
	Conformance testing or type testing	115	0.0%	306	0.2%
	Concurrent testing	21	0.0%	112	0.1%
	Alpha testing	381	0.0%	81	0.0%
	Destructive testing	1,123	0.1%	48	0.0%
	A/B testing	0	0.0%	0	0.0%
	Output comparison testing	0	0.0%	0	0.0%

Table 2: RQ2: number of matches for testing levels and testing types, techniques and tactics in the years 2015-2018.

testing is the most requested, followed by system and integration, and unit testing is the last. As expected, unit testing is explicitly required by a significant percentage of coder job advertisements (15.2%), integration testing by a mere 5%, and the other levels by a negligible rate. For testers, on the other hand, all levels but unit testing score a two digits percentage.

Concerning the "Testing Types, Techniques and Tactics", we considered 20 entries. Among them, three are by far the most popular for both Coders and Testers: regression, functional, and performance testing. After them, end-to-end scores relevantly for both Coders and Testers. Moreover, security and usability testing for Coders have the same magnitude.

The testing types, techniques, and tactics appear in a tiny percentage of coder job advertisements, with percentages up to 1.6%. On the converse, regression testing is mentioned by almost 20% of tester job ads and functional and software performance testing by 14.2% and 13.8%, respectively.

The distribution of relevance for the different kinds of testing for coders agrees with coders being more focused on software improvement than software quality assurance.

Summary. In conclusion, it emerges that the most required categories/types of software testing vary depending on the job type. The most important testing levels are unit testing for coders and acceptance testing for testers, with about the same percentage of advertisements. The most required testing types are regression, functional, and performance testing for both coders and testers. However, the relevance order for coders favors performance, then regression, then functional; for testers, on the other hand, we have regression first, then functional, then performance.

4.3 RQ3: Manual vs. Automated Testing

Procedure. To answer **RQ3**, we need to count the advertisements for jobs requiring manual/automated testing skills. Both are conceptual categories. Thus, we counted the numbers of distinct links between them and job advertisements, grouped by *O*NET* (as for the previous research question). The results are overwhelmingly in favor of automation.

To refine our analysis, we also computed the intersection of the two testing typologies. That is, we counted the advertisements mentioning both manual and automated testing. We sampled some of those ads to understand which professional figure they described. In some cases, the job offering companies looked for testers able to perform both testing kinds, in others for technical consultants to help them move from manual to automated testing.

Results. In Figure 1, both charts present the number of matches (and the corresponding percentage) for the conceptual categories of manual/automated testing. The inner rings only consider coder ads (i.e., *O*NET* 15-1132.00 and 15-1134.00). The outer rings refer to tester ones (i.e., 15-1199.0). The difference between the two charts is in the definition of *concerning automatic/manual testing*. In the *NO Tools*, we count as hits only the job advertisements tagged by the manual/automated conceptual categories. In the chart *Tools*, we extended our query for automated testing, counting also those advertisements matching some tools for automated testing, of any kind and level. We think that the latter approach is the correct one, as any advertisement requiring the use of some testing tool, implicitly refers to automated testing.

We color differently advertisements matching (1) manual but not automated testing (blue), (2) both manual and automated testing (grey), and (3) automated but not manual testing (orange).

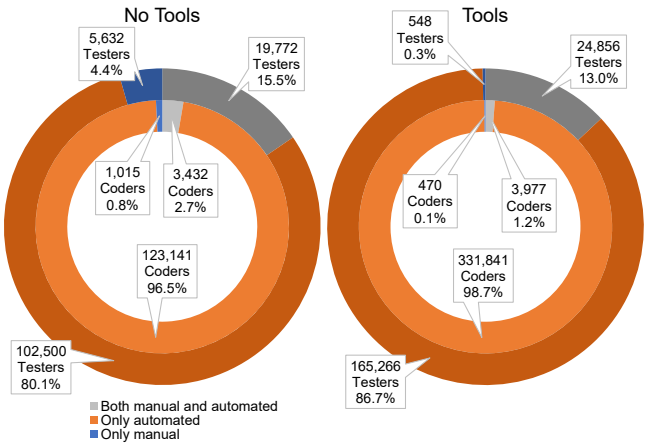

Figure 1: RQ3: advertisements for job requiring manual/automated testing skills in the years 2015-2018.

The advertisements mentioning automated but not manual testing (orange slices) cover the almost totality of those for coders (98.7% with tools, 96.5% without), and a vast majority of those for testers (86.7% with tools, 80.1% without). These results show that automated testing is by far more required than manual.

In both charts, the percentages of coder job advertisements mentioning manual but not automated testing (blue slices) are negligible (below 1%), as well as those mentioning both categories (2.7% with tools, 1.2% without). Thus, we can conclude that manual testing is not a concern for coders.

For testers, a small percentage of job advertisements mentions manual but not automated testing (0.3% with tools, 4.4% without), and a relevant part mentions both categories (13.0% with tools, 15.5% without). Thus, we can conclude that automated testing expertise is far more in demand in ads and that manual tests are a prerogative of the testers.

Summary. We can conclude that automated testing dominates the job advertisements scene since it is by far more required than manual testing for both Coders and Tester. Comparing ads mentioning just one of the two types, the ratio between the two is negligible, and even disregarding tools, it ranges from 1:20 (an advertisement concerning manual testing every 20 for automated testing) for testers to less than 1:100 for coders. Taking into account also those advertisements mentioning both categories, the ratio between the advertisements concerning automatic and manual testing ranges from 1:5 for testers disregarding tools to slightly more than 1:100 for coders considering tools.

4.4 RQ4: Testing Tools and Frameworks

Procedure. To answer **RQ4**, we ranked the conceptual categories representing testing tools on the number of matches in job advertisements. We defined two rankings, the first considering job advertisements for coders (*O*NET 15-1132.00 - Software Developers, Applications* and *15-1134.00 - Web Developers*) and the second those for testers (*O*NET 15-1199.01 - Software Quality Assurance Engineers and Testers*).

Results. Table 3 summarizes our findings for the 30-most-requested testing tools and frameworks. As in the case of **RQ2** and **RQ3**, for each entry of the table, we report the total number of matches found in (a) all the advertisements related to Coders (i.e., 15-1132.00 and 15-1134.00) and (b) all the advertisements related to Testers (i.e., 15-1199.01). For each category (i.e., Coders and Testers), the entries are in descending order of hits. The top 10 entries are highlighted with different colors in order to evidence how ranking changes when focusing on Coders and Testers only.

In the case of Coders, JUnit is by far the first, with more than 50,000 hits, followed by Selenium, with slightly more than 33,000 hits, and the pair Cucumber and Jasmine, quite outdistanced with hits in the order of 14,000.

For testers, we have Selenium in pole position with about 49,000 hits, UFT in a faraway second position with nearly 17,500 hits, and a cluster of tools, Cucumber, JUnit, and WebDriver, with hits in the order of ten thousand (about 12,000 for Cucumber and about 11,000 for the others). Note that Selenium[9] is a general term associated with a suite of tools for automating web application testing across many platforms. The two main tools of the suite are Selenium IDE and Selenium WebDriver [10]. Thus, the fifth position of WebDriver, on the one hand, clarifies its relevance among the tools within Selenium, on the other hand, makes even stronger the first position of Selenium as a whole.

[9] https://www.seleniumhq.org/

Rank	Coders		Testers	
	Framework / Tool	Number of Matches	Framework / Tool	Number of Matches
1	JUnit	50,110	Selenium	49,113
2	Selenium	33,047	UFT	17,584
3	Cucumber	14,383	Cucumber	11,980
4	Jasmine	13,911	JUnit	10,810
5	ApacheJMeter	8,343	WebDriver	10,776
6	Mocha	7,889	ApacheJMeter	8,321
7	NUnit	7,650	TestNG	6,696
8	Mockito	7,469	LoadTesting	6,118
9	TestNG	7,136	Appium	5,118
10	LoadTesting	5,898	LoadRunner	4,193
11	Protractor	5,389	Gherkin	2,688
12	SonarQube	5,276	Protractor	2,548
13	WebDriver	5,177	NUnit	2,096
14	UFT	3,885	Jasmine	1,521
15	Spock	2,739	RobotFramework	1,255
16	Jest	2,672	TestComplete	1,177
17	Rspec	2,456	Gatling	1,170
18	Appium	2,352	Watir	1,016
19	XUnit	2,093	Tosca	956
20	LoadRunner	2,028	Spock	949
21	Gherkin	1,693	Ranorex	874
22	PMD	1,524	WinRunner	727
23	AutomationAnywhere	1,464	Mocha	670
24	QUnit	1,303	RationalFunctionalTester	646
25	EasyMock	1,172	Rspec	618
26	FindBugs	1,147	XUnit	607
27	Gatling	1,118	Parasoft	576
28	Checkstyle	1,096	SilkTest	574
29	Espresso	954	SOAtest	528
30	RobotFramework	944	Calabash	526

Table 3: RQ4: Top 30 Scoring Testing Tools and Frameworks in the years 2015-2018.

Summary. Considering the overall score, and summing coder and tester hits, it emerges that the most valuable tools and frameworks are Selenium (more than 80,000 hits) and JUnit (almost 61,000 hits), followed by Cucumber (about 26,000 hits) and UFT (about 21,500 hits). In particular, Selenium is by far the most requested for Testers (even more when considered in conjunction with WebDriver) and JUnit for Coders. Besides Selenium, coder ranking favors unit testing tools, while Selenium (used for end-to-end testing) completely dominates tester ranking.

4.5 Threats to Validity

In our opinion, the main threats to validity of our study concern: (1) definition of the conceptual categories, (2) definition of the queries, (3) representativeness of the sample, and (4) geographic distribution of the advertisements.

Concerning point (1), we strived to define a comprehensive list following the procedure described in Section 3.2. That is, we integrated our experience with an analysis of authoritative sources, like classic textbooks and *Wikipedia*, and then added hits from many Google searches for covering the current practice. Moreover, one author created the list, and the others checked and extended it independently.

Concerning point (2), one author developed the queries, and the others double-checked them independently, to reduce coding errors as much as possible.

Concerning point (3), LinkUp is one of the most extensive indexes of job openings, and we analyzed a very high number of advertisements. Thus, we can reasonably assume that all kinds of IT companies are fairly represented.

Concerning point (4), the analyzed advertisements reflect the job markets where LinkUp is more used. The USA has the lion share with more than 3.3 million ads, and the faraway second is India, with less than half a million. The distribution by continent is North America 75%, Asia 13%, and Europe 9%. The other continents have negligible percentages. Thus, our study mostly focuses on the North American market. Further geographically distributed data points are highly desirable to confirm our findings worldwide. For instance, we should replicate the study with job search engines from China or Japan, preferably with the textual analysis in their local languages besides English.

5 RELATED WORKS

A Survey is a research method used for collecting data from a predefined group of respondents, usually employing questionnaires, to gain information on various topics of interest. A large number of surveys have been conducted on software testing in different countries, e.g., [5, 7–9, 17, 19].

A recent survey on software testing has been conducted in Canada with 246 practitioners [5]. The survey results reveal interesting findings on software testing practices. First, in most Canadian companies, testers are out-numbered by developers, with ratios ranging from 1:2 to 1:5. This result is in line with what we obtained in RQ1. Second, system and unit testing are two common testing types that receive the most attention and efforts, followed by GUI, acceptance, and performance testing. As reported in the sub-section concerning RQ2, our empirical study found out that unit testing is the technique most requested by the industry too. Differently, in our ranking, we have acceptance and integration testing before system testing. Maybe, this is a sign that software testing practices are changing, and nowadays, more attention is devoted to acceptance and integration testing. Third, manual testing is still in the dominant position versus automated testing. The trend is similar to another survey previously conducted in Australia [19]. Concerning this point, we obtained that companies seek much more professional competences in automated than in manual testing. However, this phenomenon could be a direct consequence of the fact that companies already have many skilled manual testers and want to invest in automated testing. Fourth, XUnit frameworks (e.g., JUnit, NUnit) and different commercial functional testing tools (e.g., IBM Rational Functional Tester) are two of the most widely used categories of testing tools. Web application testing tools (e.g., Selenium) are also widely used. This result is consistent with what we have obtained in RQ4.

Even if very common, survey research has two significant limitations compared to content analysis of job advertisements, which we conducted in our work. On the one hand, it usually works on small samples compared to content analysis. On the other hand, it treats perceived data, i.e., what a respondent believes or thinks (the perception), and not real data as content analysis often does (the reality).

A large number of content analysis studies concerning software engineering are present in literature (e.g., [1, 4, 18]). They mainly focus on technical skills, soft skills, industry needs, and trends.

Florea and Stray [4], for example, have conducted a recent content analysis study of advertisements to examine the relevance of soft skills when hiring software testers, and if there are specific skills required for agile testers. They analyzed 400 job advertisements for testers considering 33 countries and using different job-search engines. They found out that 64% of job offers require soft skills. Only 30% of the companies looked explicitly for agile testers. We share the analysis procedure with the content analysis study of Florea and Stray. Nevertheless, our goals are different and broader. Moreover, in our case, the considered sample of adverts is orders of magnitude larger.

6 CONCLUSION AND FUTURE WORK

We conducted this empirical study to take a snapshot of the industrial perceived needs in the field of software testing and use it to answer a few research questions, useful to professionals, instructors, and researchers. Our research method is content analysis, that we applied to a set of about five millions of job advertisements, taken from a popular Web job-search engine.

The main findings from the study can be summarized as follows. Software testing plays a vital role in the industry (in the adverts, the ratio between Testers and Coders is 1:6). Investing in Unit testing for Coders and in Acceptance Testing for Testers is essential. Mastering automated testing tools, such as Selenium, JUnit, and Cucumber, is beneficial for both categories. Indeed, automated testing is more requested by companies compared to manual testing.

Since this is preliminary work, we see several possible extensions. For example, (1) comparing in-depth our results with the ones obtained in related works and try to understand differences (if any); (2) conducting other experiments to confirm our data, using different job advertisement datasets, or survey-based analyses; (3) analyzing the considered RQs as a function of time; (4) extending the research questions to other detailed aspects of software testing such as end-to-end and security testing; (5) refining the analysis procedure using more in-depth and fashion approaches such as, e.g., semantic topic analysis using LDA-based topic modeling (as done, e.g., in [6]) to perfect the obtained results.

ACKNOWLEDGMENTS

We want to show our gratitude to LinkUp[10] for providing us the possibility to access to the database containing the job advertisements (i.e., the raw data).

Disclaimer. Responsibility for any information and result reported in this paper lies entirely with the Authors. LinkUp was not involved in any form of data analysis.

REFERENCES

[1] F. Ahmed, L. F. Capretz, and P. Campbell. 2012. Evaluating the Demand for Soft Skills in Software Development. *IT Professional* 14, 1 (Jan 2012), 44–49. https://doi.org/10.1109/MITP.2012.7

[2] J. L. de la Vara, M. Borg, K. Wnuk, and L. Moonen. 2016. An Industrial Survey of Safety Evidence Change Impact Analysis Practice. *IEEE Transactions on Software Engineering* 42, 12 (Dec 2016), 1095–1117. https://doi.org/10.1109/TSE.2016.2553032

[3] A. M. Fernández-Sáez, D. Caivano, M. Genero, and M. R. V. Chaudron. 2015. On the use of UML documentation in software maintenance: Results from a survey in industry. In *Proceedings of 18th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 2015)*. 292–301. https://doi.org/10.1109/MODELS.2015.7338260

[4] Raluca Florea and Viktoria Stray. 2018. Software Tester, We Want to Hire You! an Analysis of the Demand for Soft Skills. In *Agile Processes in Software Engineering and Extreme Programming*, Juan Garbajosa, Xiaofeng Wang, and Ademar Aguiar (Eds.). Springer International Publishing, Cham, 54–67.

[5] Vahid Garousi and Junji Zhi. 2013. A Survey of Software Testing Practices in Canada. *J. Syst. Softw.* 86, 5 (May 2013), 1354–1376. https://doi.org/10.1016/j.jss.2012.12.051

[6] Fatih Gurcan and Cemal Köse. 2017. Analysis of software engineering industry needs and trends: Implications for education. *International Journal of Engineering Education* 33 (01 2017), 1361–1368.

[7] ISTQB®. 2015-2016. Worldwide Software Testing Practices Report. https://www.istqb.org/documents/ISTQB_Worldwide_Software_Testing_Practices_Report.pdf, Last accessed on 2019-09-09.

[8] M. Kassab, J. F. DeFranco, and P. A. Laplante. 2017. Software Testing: The State of the Practice. *IEEE Software* 34, 5 (2017), 46–52. https://doi.org/10.1109/MS.2017.3571582

[9] Pavneet Singh Kochhar, Xin Xia, and David Lo. 2019. Practitioners' Views on Good Software Testing Practices. In *Proceedings of 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '10)*. IEEE, 61–70. https://doi.org/10.1109/ICSE-SEIP.2019.00015

[10] Maurizio Leotta, Diego Clerissi, Filippo Ricca, and Paolo Tonella. 2016. Approaches and Tools for Automated End-to-End Web Testing. *Advances in Computers* 101 (2016), 193–237. https://doi.org/10.1016/bs.adcom.2015.11.007

[11] Maurizio Leotta, Filippo Ricca, Marina Ribaudo, Gianna Reggio, Egidio Astesiano, and Tullio Vernazza. 2012. An Exploratory Survey on SOA Knowledge, Adoption and Trend in the Italian Industry. In *Proceedings of 14th International Symposium on Web Systems Evolution (WSE 2012)*. IEEE, 21–30. https://doi.org/10.1109/WSE.2012.6320528

[12] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang. 2013. What Industry Needs from Architectural Languages: A Survey. *IEEE Transactions on Software Engineering* 39, 6 (June 2013), 869–891. https://doi.org/10.1109/TSE.2012.74

[13] Patricia Mcquaid. 2012. Software disasters-understanding the past, to improve the future. *Journal of Software Maintenance and Evolution: Research and Practice - SMR* 24 (08 2012). https://doi.org/10.1002/smr.500

[14] Denise F. Polit and Cheryl Tatano Beck. 2010. Generalization in quantitative and qualitative research: myths and strategies. *International journal of nursing studies* 47 11 (2010), 1451–8.

[15] T. Punter, M. Ciolkowski, B. Freimut, and I. John. 2003. Conducting on-line surveys in software engineering. In *Proceedings of International Symposium on Empirical Software Engineering (ISESE 2003)*. 80–88. https://doi.org/10.1109/ISESE.2003.1237967

[16] Gianna Reggio, Maurizio Leotta, and Filippo Ricca. 2014. Who Knows/Uses What of the UML: A Personal Opinion Survey. In *Proceedings of 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)*, Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran (Eds.). LNCS, Vol. 8767. Springer, 149–165. https://doi.org/10.1007/978-3-319-11653-2_10

[17] Laura Strazdina, Vineta Arnicane, Guntis Arnicans, Janis Bicevskis, Juris Borzovs, and Ivans Kulesovs. 2018. What Software Test Approaches, Methods, and Techniques are Actually Used in Software Industry?. In *Proceedings of Baltic DB&IS 2018 Conference Forum and Doctoral Consortium*, Vol. 2158. CEUR Workshop Proceedings, 13–22.

[18] Sami Surakka. 2005. Analysis of Technical Skills in Job Advertisements Targeted at Software Developers. *Informatics in Education* 4 (2005), 101–122.

[19] Ossi Taipale, Kari Smolander, and Heikki Kälviäinen. 2005. Finding and Ranking Research Directions for Software Testing. In *Software Process Improvement*, Ita Richardson, Pekka Abrahamsson, and Richard Messnarz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 39–48.

[20] Marco Torchiano, Federico Tomassetti, Filippo Ricca, Alessandro Tiso, and Gianna Reggio. 2013. Relevance, benefits, and problems of software modelling and model driven techniques—A survey in the Italian industry. *Journal of Systems and Software* 86, 8 (2013), 2110 – 2126. https://doi.org/10.1016/j.jss.2013.03.084

[21] Marilyn White and Emily Marsh. 2006. Content Analysis: A Flexible Methodology. *Library Trends* 55 (06 2006). https://doi.org/10.1353/lib.2006.0053

[22] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.

[10] https://www.linkup.com/