Dissecting the secrets of software testing education in universities

Onur Gökmen[1]

Supervisor(s): Andy Zaidman[1]**, Baris Ardıç**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Onur Gökmen
Final project course: CSE3000 Research Project
Thesis committee: Andy Zaidman, Baris Ardıç , Koen Langendoen

Abstract

Software testing plays a crucial role in delivering reliable software. Currently, research is ongoing on how software developers and testers acquire this knowledge of software testing to deliver reliable software and what kind of knowledge is being transferred to developers and testers. In an effort to gain more insight into this area, we will focus on answering which software testing topics are being discussed in dedicated software testing courses and software engineering courses in top-ranked universities. Our findings show us that white-box testing, black-box testing and the discussion of test levels are the most commonly discussed topics in universities.

1 Introduction

In the current society, software is used for a wide range of activities. Software helps to solve difficult problems, speeds up information processing, automates repetitive tasks and much more. To develop well-functioning, robust and reliable software, software is being tested rigorously in different phases of development [1].

Developers and testers play a crucial role in delivering quality software by conducting various software testing techniques [2]. However, software is becoming more and more complex and testers must identify and simulate services that software systems use and test these services [3]. Software that is not properly tested may lead to disastrous events, such as causing economic losses and even human deaths [4].

To prevent these disasters, developers and testers shared in a survey conducted by JetBrains that software testing is increasingly becoming an integral part of their development [5].

On top of that, Augusto et al. analysed the impact of software testing education in higher education on code reliability and concluded that developers who learn basic testing principles are more than twice likely to produce reliable code [6].

It is clear that software testing is an integral part of the software development life cycle and gaining software testing knowledge results in reliable code. This highlights the importance of investigating which software testing topics are being discussed in software engineering and software testing courses. Thus, the main research question is formulated as follows:

What major software testing topics are being discussed in syllabi of software engineering and software testing courses at university level? In order to address our main research question, we will divide it into four smaller parts.

1. What are the course contents of software testing and software engineering courses?
2. What are the learning objectives of software testing and software engineering courses?
3. What are the course activities of software testing and software engineering courses?
4. How do course contents compare in different continents?

By exploring university courses that teach about software testing techniques, we will gain insight into what testing techniques are taught to students and the latest trends regarding software testing in universities. Moreover, similarities and differences in software testing topics and course activities will be discovered by comparing continents.

The structure of this research is as follows: In Section 2, we first explore related work to our main research question. Section 3 describes our methodology, Section 4 details the findings of our research. In Section 5, we will discuss these results. After that, in Section 6 we will discuss the reproducibility and ethical aspects of this research and finally, we conclude our research and propose future research topics in Section 7.

2 Related Work

Currently, researchers and educators are working on analyzing and improving software testing education. Additionally, systematic literature review and systematic literature mapping studies are conducted to provide an overview of the research landscape regarding software testing and software engineering fields. Finally, the importance of software testing knowledge is highlighted in some of the research papers. This section will state research papers that have contributed to these categories.

Garousi and Mathur provided an overview of the state of software testing education by researching course contents and projects in top universities in Canada and North America [7]. Course projects were not up-to-date with the most recent testing tools and technologies. In addition, most of the projects are not built on real-world Systems Under Test, but rather used "toy" examples to teach software testing topics.

On top of the overview of Canada and North America, Ardic and Zaidman analyzed dedicated software testing courses in top universities according to the Times ranking and concluded that different testing techniques, test design and planning were the most popular testing skill categories [8].

Garousi et al. analyzed the knowledge gap between software engineering courses and industrial needs by reviewing papers related to software engineering courses [9]. They state that the biggest knowledge gaps are in requirements engineering, design and testing. The authors implied that teaching materials in software engineering courses are not aligned with Industry's needs.

Garoushi et al. have conducted a systematic literature mapping on research papers associated with software testing education [10]. The results show that software testing education is becoming more active, which is concluded by the increasing number of written papers regarding software testing education.

Finally, the importance of software testing knowledge is emphasized by Augusto et al [6]. University students implemented two different functions before and after learning software testing concepts like black-box, white-box and mutation testing. Students with software testing knowledge are more than twice likely to produce reliable code, than students that did not receive software testing knowledge before implementation.

3 Methodology

To answer our research questions, we first selected specific software engineering and software testing courses. Subsequently, we collected data regarding these courses and finally, we performed manual keyword extractions and grouping techniques from the data that we collected.

3.1 Software Engineering and Software Testing courses

In this study, we focused on software engineering and software testing courses. These courses may overlap in the teaching material that is taught, as software testing is a subset of software engineering. In this study, we defined software testing courses as courses that primarily focus on teaching software testing topics. Henceforth, we will refer to software testing courses as dedicated software testing courses.

On top of software testing topics, software engineering courses also teach about designing, building and maintaining software. In this study, we omitted these topics as they are out of the scope.

3.2 Course selection

To initiate our research, we explored the top 150 universities in the subject of Computer Science and Information Systems by QS World University Rankings [11]. In this study, the importance of the teaching quality outweighs other indicators. Therefore, we opted to use QS ranking, because the primary indicator for the construction of the ranking is academic reputation and teaching methods.

In order to find the course syllabus and catalogue, we used a list created by Ardic and Zaidman [8]. This list already provides the course syllabus for software testing courses and links to the course catalogue for software engineering courses in the top 100 based on the Times Higher Education ranking. For courses that are not present in this list, we manually searched the course catalogue. The following keywords were used in search engines:

- "University name + course catalog"
- "University name + module catalog"
- "University name + Computer science courses"
- "University name + Computer science course list"

For course catalogues that were not found using this approach, we manually searched university websites and the related department regarding computer science, software engineering and information systems to find course catalogues or the curriculum of the study program.

Next, we analyzed course descriptions and contents to determine the categorization of a course as a dedicated software testing course or as a software engineering course. Courses that primarily teach about software testing are classified as dedicated software testing, whereas courses containing "software" in their title, with the addition of "software testing" or "testing" in the course descriptions and contents are classified as software engineering courses.

3.3 Data collection and Labelling

The course syllabi functioned as our primary source of data to potentially find information regarding course content, course activities and learning objectives. The details of the course syllabus ranged from comprehensive information with bullet points regarding the discussed topics, to limited available information, providing only a few sentences with high-level keywords like software testing and quality assurance.

Besides the course syllabus, we also attempted to find additional information, which mostly originated from course website, the professor's websites and Github links. Keywords like "University name + Course code", "University name + Course code + Professor name" and "University name + Course code + Github" were used in search engines. We then browsed through every page of the search, which mostly consisted of 5-10 pages at maximum. Occasionally, obtaining additional information from previous years was more accessible. Subsequently, modifying the course year to the current year in the URL occasionally provided access to recent information.

Once we had access to course information, we manually labelled the course contents, course activities and learning objectives. These learning objectives were then grouped into categories. Software testing courses have categories which are similar to the categorization of learning outcomes by Ardic and Zaidman with the addition of 5 more categories [8]. For software engineering courses, however, we have created our own categories. We first extracted keywords for course objectives and later we merged these keywords into groups. Our categories represent a high overview of learning objectives.

3.4 Data set

By utilizing this method, we have gathered 212 entries from which 79 are dedicated software testing courses, 91 are software engineering courses and the remaining 42 entries were universities that are not classified, because no information could be found or no courses provided software testing education. Therefore these 42 entries are no longer used in the rest of this study.

Our investigation covered 61 courses in Europe, 59 courses in North America, 34 courses in Asia, 12 courses in Australia, and 4 courses in Latin America. 100 courses were part of a bachelor's program, whereas 70 courses were part of a master's program. In total 56 courses provided additional information, such as slides, lecture videos and information regarding learning objectives and course activities on course websites, Github links or an educator's website. The data collected in this study is made available on our Github page [12].

4 Results

In this section, we will present our findings regarding course contents, course activities and learning objectives.

4.1 Software testing topics

Dedicated software testing courses

The general overview of software testing topics in dedicated software testing courses is displayed in Figure 1. During

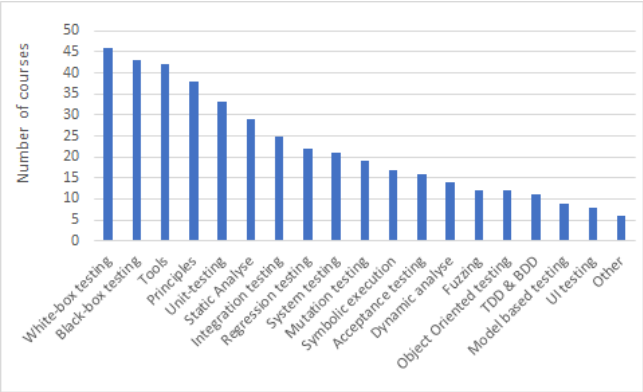

Figure 1: Overview of software testing topics in dedicated software testing courses.

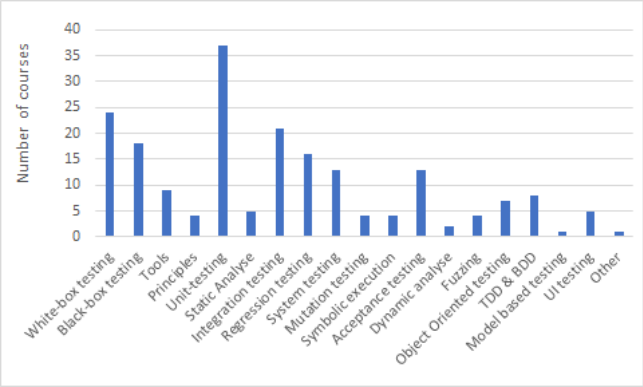

Figure 2: Overview of software testing topics in software engineering courses.

our investigation, we have perceived that the details of syllabi can vary substantially. Some universities provided detailed information, which made collecting course content easier. But, more often, syllabi contained high-level concepts such as black-box testing and white-box testing. Therefore, we have categorized the keywords such as coverages, control and data flow to white-box testing and partitioning, boundary and combinatorial testing to black-box testing.

The most taught topics were white-box testing, black-box testing and testing tools, which all occurred in more than 40 out of 79 courses. The discussion of test levels, like unit testing, integration testing, system testing and acceptance testing also occurred relatively frequently.

In the category "Other", we combined testing techniques such as property testing, metamorphic testing, search-based testing, security testing and syntax testing, which occurred in less than 7 courses. We also observed that symbolic execution is commonly taught along with formal verification methods and fuzzing. These topics were mostly taught in graduate programs. In total, 14 courses taught about symbolic execution and 11 courses taught about fuzzing in graduate programs. However, in undergraduate programs 3 courses taught about symbolic execution and 1 course about fuzzing.

Finally, 13 universities provided a syllabus, but did not provide enough details to extract course information. The contents of these syllabi were not extracted in this study.

Software engineering courses

Figure 2 contains an overview of each software engineering course's contents.

As shown in the table, the most covered topic was unit testing, followed by testing techniques like white-box testing, integration testing and black-box testing. The rest of the test levels in combination with regression testing also appeared in course syllabi.

Upon comparing Figure 2 to Figure 1, it becomes clear that the discussion of testing techniques defined in the category "Other", model-based testing, mutation testing, symbolic execution and fuzzing are not emphasized in software engineering courses. Finally, 39 software engineering courses mentioned that software testing is part of the course contents, but did not provide information on what topics are being discussed.

4.2 Course objectives

Dedicated software testing courses

In total, 48 out of 79 courses provided learning objectives in course syllabi. In addition, we also considered course descriptions if and only if the course descriptions mentioned information regarding learning objectives. Using the description, a total of 10 courses were classified.

To classify and group the learning objectives, we have used the categories that are introduced by Ardic and Zaidman [8]. This list categorizes common learning objectives found in course syllabi related to software testing. We also introduced 5 additional categories, because not all learning objectives fit in these categories. The most frequent learning objective categories with the number of occurrences are listed as follows:

- **Practical experience (36)** - Assess students in terms of writing reliable test code by avoiding bad programming practices in testing and by implementing test techniques.
- **Testing techniques (27)** - Assess the understanding and knowledge of various software testing techniques.
- **Tools and frameworks (25)** - Assess the knowledge and understanding of software testing tools and frameworks.
- **Fundamentals of software testing techniques (23)** - Assess the fundamentals of software quality and software testing, in terms of having basic knowledge regarding concepts, models and terms.
- **Testing strategy and plans (20)** - Select and design test strategies, plans and cases.
- **Automated testing (13)** - Assess the knowledge and understanding of test automation and analysis.
- **Research (11)** - Explain current trends in software quality and testing research.
- **Quality assurance techniques (10)** - Students are able to explain the properties, strengths and weaknesses of quality assurance techniques.

- **Proper technique (10)** - Justify the use of a software testing technique.
- **Conduct quality control (10)** - Students are able to conduct quality control techniques such as test reviews, test inspections, code refactoring and analyze test reports.
- **Measurements (8)** - Measure the efficiency of test suites for software. In addition, students are able to apply measurement techniques.
- **Coverage (8)** - Students are able to recall different test coverage.
- **Evaluate test process, techniques and outcomes (7)** - Students are able to assess, evaluate and compute test process, techniques and outcomes.
- **Metrics (7)** - Students are able to describe and apply software metrics.
- **Select appropriate technique (3)** - Students are able to select and apply appropriate methods or techniques to build quality and dependability into software systems.

The constructed list indicates that educators emphasize practical experience. Moreover, the ability to understand and apply various testing techniques deem important. Finally, the understanding and usage of testing tools and frameworks alongside grasping the fundamentals of software testing techniques are also expected from students.

Overall, this implies that at the end of the courses, students should have solid fundamentals of software testing, be able to apply and implement different testing techniques and be able to use tools and frameworks.

Software engineering courses

In total, 53 out of 91 courses provided learning objectives in course syllabi. We also used course descriptions to identify learning objectives. A total of 21 courses were categorized using this approach.

In general, we have observed that software engineering courses contained different topics, such as the principles of software engineering and development, architecture and design, software requirements and testing. Courses provided minimal depth and information related to these learning objectives, often just mentioning these keywords. Thus, our list of categories provides a high overview of these learning objectives. The list of categories with the number of occurrences is demonstrated as follows:

- **Software engineering and development (60)** - Understand and practise the principles of software engineering and development.
- **Software testing (44)** - Understand and apply software testing techniques.
- **Software architecture and design (42)** - Students are able to explain and apply different software architectures and designs.
- **Software management and maintenance (30)** - Assess the understanding and knowledge of software management and maintenance techniques.

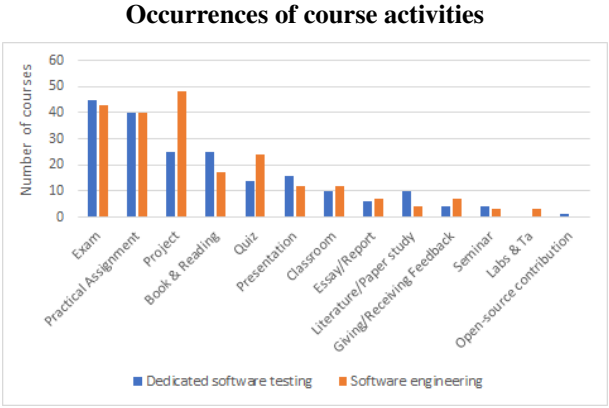

Figure 3: Overview of course activities in software engineering and dedicated software testing courses.

- **Software requirements (27)** - Students are able to carry out requirement analysis and are able to write requirements definitions.
- **Large scale and complex software development (18)** - Students are able to produce complex software with industrial strength quality.
- **Theoretical approach to software testing (11)** - Assess the understanding and knowledge of software testing, verification and validation techniques.
- **Practical approach to software testing (10)** - Students are able to apply software testing techniques in code bases.

The constructed list indicates that software testing is an important factor regarding learning objectives. The importance of the theoretical aspect of software testing is highlighted in 11 courses, whereas 10 courses emphasized the practical side of software testing. Both categories are a subset of the software testing category.

4.3 Course activities

The outline of course activities in software engineering and dedicated software testing courses are depicted in Figure 3. An interesting observation is that software engineering and dedicated software testing courses provide students with practical learning experiences through assignments or group projects. This corresponds with the practical experience objective, which was introduced in Section 4.2. Moreover, the following observation were found:

- In total 14 courses a combined project with a presentation.
- There are 11 courses that did not have an exam, but mainly assessed learning objectives through quizzes. These courses are all except for one originating in North America. 24 courses provided an exam together with quizzes.
- Literature research is often combined with presentations or a report/essay, which occurs in 10 out of 14 courses.

In terms of course activities, we observed that dedicated software testing courses are quite similar to software engineering courses. The main difference is that software engineering courses incorporate projects even more than exams.

5 Discussion

In this section, we summarize the results and answer our research questions.

RQ1: *What are the course contents of software testing and software engineering courses?* In dedicated software testing courses, white-box testing, black-box testing, tools regarding software testing and unit testing are the most discussed software testing topics. We also observed that topics, such as symbolic execution, fuzzing, model-based testing, security testing, metamorphic testing and automated test case generation frameworks like Randoop and Korat are also present in course syllabi. However, these topics rarely appear and are mostly found in master's studies.

In software engineering courses, unit testing, white-box testing, integration testing and black-box testing are the most taught testing topics. The application of unit and integration testing are integral parts of software testing as unit testing validates individual components of software, whereas integration testing validates the interaction of different software systems. Both of the testing techniques are well-suited to combine with software projects, which is the main form of assessment in software engineering courses.

RQ2: *What are the learning objectives of software testing and software engineering courses?* In dedicated software testing courses, practical experience, test techniques, and tools and frameworks are the most common learning objectives.

In software engineering courses, excluding the principles of software engineering and development, the most common learning objective is software testing, which occurred in 48 % of the courses. As such, we see a strong indication that educators consider software testing as an important aspect of software engineering. Different topics, such as software architecture, design and requirements are also present in software engineering courses.

RQ3: *What are the course activities of software testing and software engineering courses?* In dedicated software testing courses, educators prefer assessing students through exams, which occur in 45 out of 79 courses. In addition, educators provide practical experience to students by either assigning them a group project or by providing practical exercises related to software testing. In total, 40 courses provided practical assignments and 25 courses provided projects.

Upon comparing the number of occurrences of practical assignments or projects across continents, we observed that 25 out of 30 courses in Europe provided practical experience, 15 out of 24 courses in North America, 7 out of 13 courses in Asia, 4 out of 8 courses in Australia and finally 2 out of 4 courses in Latin America. These numbers imply that practical assignments or projects are incorporated around the globe.

Reading materials and quizzes are also incorporated in dedicated software testing courses. There are 14 courses that provided quizzes as a form of assessment, of which 6 are in

Comparison of course content

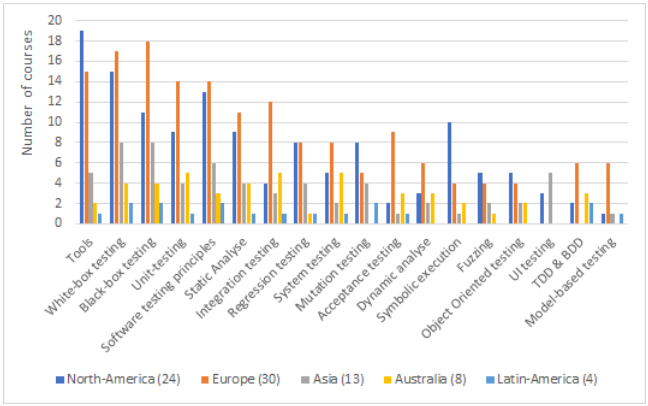

Figure 4: Overview of course contents in dedicated software testing courses across the continents. The number after the continent in the x-axis represents the number of courses in the respective continent.

North America, 4 in Asia, 2 in Europe and 2 in Australia.

In software engineering courses, educators mainly conveyed the learning objectives and the course contents through a group project, which occurred in 48 out of 91 courses. Practical assignments are also used to convey practical experience to students, which is used in 40 out of 91 courses. In contrast, Figure 3 shows us that exams are incorporated less than projects. We observed that North America consistently provides students with group projects, namely 25 out of 35 courses. In Europe, 16 out of 31 courses, in Asia 6 out of 21 courses and finally, in Australia 1 out of 4 courses provided a project.

RQ4: *How do course contents compare in different continents?* The overview of the course contents of dedicated software testing courses in different continents is listed in Figure 4. From this figure, it becomes clear that black-box testing, white-box testing and tools are discussed globally since they appear the most in each continent. Test levels, like unit testing, integration testing, system testing and acceptance testing, are consistently discussed in Europe. In other continents, the discussion of test levels is not as quite common as in Europe. For example, the number of occurrences of integration testing, system testing and acceptance testing is almost similar in North America, Asia and Australia. In addition, model-based testing, TDD and BDD are discussed more frequently in Europe.

On the other hand, symbolic execution demonstrates a greater occurrence in North America, when compared to the rest of the continents. Finally, the discussion of UI testing is more common in Asia than in the rest of the continents.

Figure 5 contains an overview of course contents of software engineering courses across the continents. In general, limited information are available regarding course contents. Courses provided general terms like software testing, software quality and testing in course contents. So deducing software testing topics, in this case, is not possible. In North America, for 24 courses out of 35, we had access to additional information, like course slides, lectures and Github links of

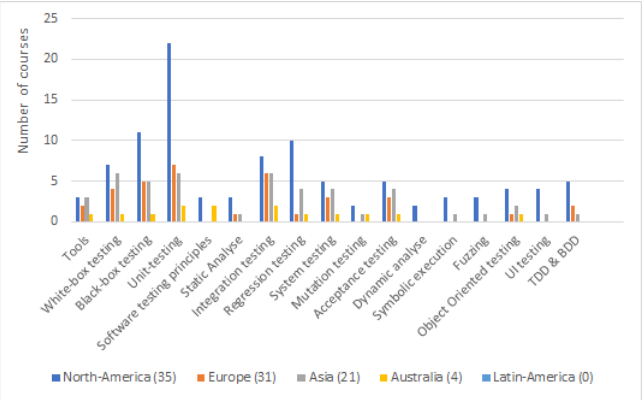

Figure 5: Overview of course contents in software engineering courses across the continents. The number after the continent in the x-axis represents the number of courses in the respective continent.

courses. These sources contained detailed information related to course contents and thus extracting information in North America was more convenient than in the rest of the continents. In the rest of the continent, we, unfortunately, found around 3 to 4 courses providing additional information. From Figure 5, we can confidently conclude that in all continents, unit and integration testing are the most taught topics. Black-box, white-box testing techniques and regression testing also appear in course contents.

6 Responsible Research

In an effort to make this research as reproducible as possible, we have provided the course catalogues, course syllabi and additional information links of courses that are considered in this data package. We have also provided links to course catalogues when we did not find any software testing courses and software engineering courses that did not teach about software testing topics. We have observed that course syllabi are not complete, because not all universities provide detailed information in syllabi. In order to mitigate this risk, we have made an effort to find additional information related to these topics. All in all, we can not be entirely sure if a course does or does not address software testing topics, learning objectives or course activities.

In this study, we have mainly collected our courses via QS ranking [11]. There exist different rankings, each using different indicators to rank universities. Depending on the ranking, the consideration of universities will change and therefore the data package and the results will change. Moreover, since we have used a ranking, based on academic reputation and teaching quality, we are aware that the ranking will contain more universities in Europe and North America, than in the rest of the continents. In addition, we have observed that information in these continents is more available than in Asia, Australia and Latin America. This might skew the findings of this study in favour of these continents. In order to gain a full overview of the continents, further region-based investigations are needed and universities should provide more information regarding courses.

We have utilized open coding as a methodological approach to categorize course contents and course activities. Although, there are subjective elements involved in open coding, we have only extracted keywords depending on what is available to us regarding course contents and activities. For learning objectives in software testing courses, We have mainly used categories defined by Ardic and Zaidman [8]. In addition, We have merged keywords that could not fit into these categories into new groups, leading to 5 more categories. In a similar way, we have also categorized learning objectives for software engineering courses. We first identified learning objectives and in the next steps, we have combined related learning objectives into new categories. Since the categorization of the keywords is subjective by nature, these categories may differ when undertaken by another person.

7 Conclusions and Future Work

This study aimed to find out which software testing topics are being discussed in software engineering and software testing courses in top-ranked universities. The main observations of this research were that white-box testing, black-box testing and testing tools are the most commonly taught topics in software testing courses. In software engineering courses, unit testing, white-box testing, integration testing and black-box testing are the most discussed topics. Furthermore, in this study, we also address the learning objectives and course activities of these courses.

In this study, the availability of information in course syllabi formed a limitation, which differs heavily in each continent regarding course contents, learning objectives and course activities. Thus our findings are not complete, but rather provide a general overview of each continent with the available information.

In future work, this study can be used to compare different learning resources related to software testing education, such as MOOCs, books and the Internet. Furthermore, this work can also be used by educators to gain insights into the topics being covered by course syllabi, enabling them to structure their own syllabi.

References

[1] Okeke Stephen and K Oriaku. Software development methodologies: Agile model vs v-model. *International Journal of Engineering and Technical Research*, 2:108–113, 11 2014.

[2] Mauricio Aniche. *Effective software testing : a developer's guide*. Manning Publications, Shelter Island, NY, 1 edition, 2022.

[3] J.A. Whittaker. What is software testing? and why is it so hard? *IEEE Software*, 17(1):70–79, 2000.

[4] Theo Leggett. 737 max crashes: Boeing says not guilty to fraud charge. *BBC News*, 2023. available: https://www.bbc.com/news/business-64390546.

[5] Jetbrains survey testing 2022. available: https://www.jetbrains.com/lp/devecosystem-2022/testing/.

[6] Otavio Augusto Lazzarini Lemos, Fábio Fagundes Silveira, Fabiano Cutigi Ferrari, and Alessandro Garcia. The impact of software testing education on code reliability: An empirical assessment. *Journal of Systems and Software*, 137:497–511, 2018.

[7] Vahid Garousi and Aditya Mathur. Current state of the software testing education in north american academia and some recommendations for the new educators. In *2010 23rd IEEE Conference on Software Engineering Education and Training*, pages 89–96, 2010.

[8] Baris Ardic and Andy Zaidman. Hey teachers, teach those kids some software testing. *Proceedings of the Fifth ICSE Workshop on Software Engineering Education for the Next Generation (ICSE SEENG)*, pages 9–16, 2023.

[9] Vahid Garousi, Gorkem Giray, Eray Tuzun, Cagatay Catal, and Michael Felderer. Closing the gap between software engineering education and industrial needs. *IEEE Software*, 37(2):68–77, 2020.

[10] Vahid Garousi, Austen Rainer, Per Lauvås, and Andrea Arcuri. Software-testing education: A systematic literature mapping. *Journal of Systems and Software*, 165:110570, 2020.

[11] Qs world university rankings by subject 2023: Computer science and information systems. available: https://www.topuniversities.com/university-rankings/university-subject-rankings/2023/computer-science-information-systems.

[12] Onur Gökmen. Study data, 2023. available: https://github.com/onurgkmn/CSE3000.