






State of the Practice in Software Testing Teaching in Four European Countries


Porfirio Tramontana 
University of Naples Federico II
Naples, Italy
ptramont@unina.it

Beatriz Marín 
Universitat Politècnica de València
València, Spain
bmarin@dsic.upv.es


Ana C. R. Paiva 
University of Porto & INESC TEC
Porto, Portugal
apaiva@fe.up.pt


Alexandra Mendes 
University of Porto & HASLab / INESC TEC
Porto, Portugal
alexandra@archimendes.com

Tanja E. J. Vos 
Universitat Politècnica de València
Open Universiteit, The Netherlands
tvos@dsic.upv.es

Domenico Amalfitano 
University of Naples Federico II
Naples, Italy
domenico.amalfitano@unina.it

Felix Cammaerts 
KU Leuven
Leuven, Belgium
felix.cammaerts@kuleuven.be

Monique Snoeck 
KU Leuven
Leuven, Belgium
monique.snoeck@kuleuven.be

Anna Rita Fasolino 
University of Naples Federico II
Naples, Italy
fasolino@unina.it

Abstract—Software testing is an indispensable component of software development, yet it often receives insufficient attention. The lack of a robust testing culture within computer science and informatics curricula contributes to a shortage of testing expertise in the software industry. Addressing this problem at its root —education— is paramount. In this paper, we conduct a comprehensive mapping review of software testing courses, elucidating their core attributes and shedding light on prevalent subjects and instructional methodologies. We mapped 117 courses offered by Computer Science (and related) degrees in 49 academic institutions from four Western European countries, namely Belgium, Italy, Portugal and Spain. The testing subjects were mapped against the conceptual framework provided by the ISO/IEC/IEEE 29119 standard on software testing. Among the results, the study showed that dedicated software testing courses are offered by only 39% of the analysed universities, whereas the basics of software testing are taught in at least one course at every university. The analysis of the software testing topics highlights the gaps that need to be filled in order to better align the current academic offerings with the real industry needs.

Index Terms—Software testing, Testing education, Testing Teaching, Mapping of courses

I. INTRODUCTION

As the importance of software in our society continues to grow, the impact of software failures becomes more significant [1], [2]. In the US alone, the cost of poor software quality has been estimated at \$2.08 trillion for 2020 [3]. Although software testing is a critical component in ensuring the quality of software and reducing the risks associated with software failures, it is still often overlooked. There have been efforts to mitigate the software failures reported by Krasner [3], such as the Common Vulnerabilities and Exposures (CVE) [4] which aims to catalogue publicly known vulnerabilities and exposures in software. Despite this, a report by Failwatch

has still identified 606 software failures affecting 3.6 billion people [1].

The industry faces several testing challenges related to test case design, scripting, execution, reporting, management, automation and even the lack of general knowledge [5], [6], [7]. There is also a lack of testing culture in organisations. Programmers may understand the importance of testing, but they often put it off because of the pressure to deliver quickly [7]. In addition, the quality of test cases has been found to be influenced by the domain knowledge and testing expertise of the person performing the testing process [6].

We advocate that the problem should be tackled at its root: education. However, software testing is often neglected in computer science courses. For example, curricula spend more time on more ‘glamorous’ topics [8], such as artificial intelligence.

At university level, several efforts have been made to improve teaching techniques for testing in order to better prepare students for industry, as reported by [9] and [10]. Nevertheless, there are still many problems with teaching testing, such as the disconnection between theory and practice, which leads to less interest on the part of students; classroom examples are far removed from real-world projects, or they are focused on a particular testing technique; students’ lack of testing experience may lead to them not being able to perform the testing process with all the steps in industry; students are not confident in their testing skills, etc.

Therefore, when seeking a solution to this problem in education it is important to consider three perspectives: the needs of students, academia, and industry [11], [12]. Regarding industry, a categorisation of industry needs for testing should be identified. For academia, an instructional design for

early/seamless teaching testing materials should be developed. Regarding students, the cognitive models of students during the practice of software testing should be taken into account. Taking these three perspectives together, teaching capsules can be proposed that aim at early, seamless integration into education with appropriate instructional design.

This paper takes a deeper look at the academic perspective by investigating the state of the practice in software testing education in academic institutions. This will allow for a more seamless integration of educational software testing tools into current curricula. There are some works that have already investigated the state of software testing education in higher education [13], [14], [15], [16]. However, these works focus on individual countries or on the world as a whole, i.e. none of the existing studies focus on the European context. As a result, a more general overview of the state of software testing education in Europe is lacking in the literature.

To fill this gap, we decided to carry out a study investigating the state of the practice in software testing education in academic institutions in the context of four Western European countries. We conducted this study as part of the ENACTEST Project [17] which aims to improve the current practices in software testing education. Indeed, one of the goals of the ENACTEST Project researchers is to propose new software testing teaching materials, the so-called capsules, that will be aligned with the industry and students' learning needs [12].

In this paper we present the results of the study where we focused on the higher education courses offered by academic institutions belonging to Belgium, Italy, Portugal, and Spain. We chose these countries for convenience, since they coincide with the countries of four of the partners of the ENACTEST Project. The aim of the study is to understand the diffusion of software testing-related courses in the considered context, the aspects of software testing that are most commonly taught, and course-specific teaching characteristics including Educational Level, Number of credits, Duration, Student Assessments, Testing topics taught, and Reference books. Thus, the contributions of this paper are: (1) the current state of testing education practice in the considered European countries, (2) the topics that are commonly taught, and (3) the identification of improvements needed for testing education in computer science or informatics curricula.

The contribution of this work is useful for academics, researchers, and practitioners. Academic institutions can use our results to improve their relevant curricula by adding testing topics that are commonly taught in these European countries. Practitioners can use the information provided here to prioritise training on specific topics that are not usually taught in testing courses. Researchers can use our results to develop new approaches that can reduce teachers' workload in teaching testing and improve students' learning effectiveness.

The paper is organised as follows. Section II presents related works that investigate the state and diffusion of testing courses in computer science curricula. Section III illustrates the aim of our study, with the corresponding research questions, target population, and data collection procedures. Section IV presents

the results of our study and discusses threats to validity. Section V provides a discussion of the most commonly taught testing topics, the reference books used in testing courses, and future research directions. Finally, Section VI presents our main conclusions and future works.

II. RELATED WORK

A systematic mapping study conducted in 2019 [9], has analysed 293 papers on the integration of software testing in introductory programming courses. The study reports that research on these topics can be divided into several categories, namely, teaching methods, course materials, programming assignments, programming process, program/test quality, concept understanding and tools. The study also provided several benefits and drawbacks on the integration of software testing into programming courses. Benefits include: timely feedback, objective assessment and improvement in students' programming performance. Drawbacks include additional workload for course staff, student's reluctance to conduct testing and programming courses already being packed. Nevertheless, this work does not provide evidence of the number of courses that include testing topics in programming courses, neither the institutions that are offering the courses.

A survey published in 2010 [13], conducted among randomly chosen universities in Canada and the US revealed that many computer science degree programs did not include dedicated courses for teaching software testing (ST) in their curricula. The findings indicated that two out of the top five Canadian universities and seven out of ten universities in the US did not offer standalone software testing courses.

A systematic mapping of the literature on testing education was published in 2020 [10], which was focused on exploring approaches to improve testing education. This literature review analyzes 204 papers from 1992 to 2019. The results show that there are several approaches to improve software testing education, which have been evaluated either in specific testing courses or integrated in non-specific testing courses. Nevertheless, the authors recognize that the type of testing activities performed in the courses are a very small sub-set of courses taught at universities since educators usually do not publish the organization properties of testing courses. Moreover, a clear characterization of the software testing courses and the corresponding universities is missing in this work.

The state of undergraduate software testing education in Brazil in 2012 was published in [14]. This study involved a comparison of the course recommendations provided by the Brazilian Computer Society (SBC) with the curricula of 25 Brazilian universities. Additionally, the study extended its analysis to include 21 international universities from various countries, including the United States (13), the United Kingdom (3), Switzerland (2), China (1), the Netherlands (1), and Singapore (1), in which the course recommendations of the Association for Computing Machinery (ACM) were considered. In both the Brazilian and international university contexts, the analysis revealed a common issue: there was an insufficient allocation of lectures dedicated to teaching

software testing, indicating that software testing practices were not adequately covered.

A replicated study was performed in 2020 [18]. This research involved a survey of courses that incorporated topics related to Software Testing in 28 Brazilian universities. Their findings revealed that specific courses dedicated to Software Testing were offered in 68% of the universities. However, it was noted that, with the exception of only two universities, such courses were optional rather than mandatory.

In a more recent study [19], a global perspective on how instructors addressed the topic of Software Testing was performed. Although global, it is worth noting that Africa was not included. The study focused on various aspects, such as course content, teaching methods, the use of educational resources, and examination methods for students. One of the noteworthy findings from this study was the presence of commonalities in the subjects covered. Specifically, functional testing emerged as the most frequently taught topic. Additionally, their study revealed that the traditional approach of classroom-based teaching was the prevailing method used in this context.

A systematic survey of syllabi for courses related to software testing offered in Sweden in 2022 was presented in [16]. This study examined course offerings from 25 Swedish universities that provided degrees in Computer Science or related fields. Their findings indicated that among these universities, 14 currently provide specific courses in software testing. Furthermore, they observed that approximately 32% of these individual courses were available at the undergraduate level. Moreover, about 28% of the universities offered courses aimed at specialized training in testing. In the majority of the universities surveyed, dedicated software testing courses made up roughly 5% of the total degree credits offered.

A recent study [15] investigated the curricula of 100 highly ranked universities in Asia, America and Europe. Results in this study indicates that just half of computer science curricula has software testing courses. However, a deep analysis of the specific topics of testing taught in these courses is lacking.

Despite the importance of having practical knowledge of courses about software testing with their main characteristics, the related work reveals that there isn't a clear characterization of software testing courses in the context of Europe. We address this gap in this work by performing a mapping review of highly ranked academic institutions of a set of representative countries in Europe.

III. STUDY DESIGN

In this study we were interested in knowing the state of the practice of teaching software testing in academic institutions from four European countries, namely, Belgium, Italy, Portugal and Spain. Although other types of institutions, like higher-education schools and professional bachelors, also offer curricula with software testing, we did not consider them since they were out from the scope of this study.

We used the Goal-Question-Metric template [20] to define the goal of the study as follows: *Analyse* software testing courses at the academic level and their characteristics *for*

the purpose of understanding the state of the practice *with respect to* software testing education *from the point of view of* researchers and professors *in the context of* ranked universities in European countries.

A. Research Questions

In order to achieve the goal of our study, we have formulated the following research questions:

- RQ1 How common are software testing related courses in the considered academic context?
- RQ2 What are the educational organisational properties of these courses?
- RQ3 What aspects of software testing are most commonly taught?

The first question aims to provide an overview of the courses that focus entirely on Software Testing (ST), and courses that include other topics but also Software Testing topics (NST) in computer science related degrees.

The second question is designed to characterise the courses on the basis of their educational organisational properties. This question aims to provide an overview of whether these courses are offered at either Bachelor or Master level in the academic institutions considered. In addition, this question aims to provide information on teaching methods (theoretical or practical) and the course characteristics, i.e. the course name, year, the number of credits, duration in terms of hours, assessment methods, and reference books.

The last question aims to investigate the topics that are more often taught in the mapped courses and also the ones that are missing in those courses. To properly identify the testing topics, we use the internationally agreed ISO/IEC/IEEE 29119 series of standards for software testing. We chose this series of standards because they are intended to be used by any organization when performing any form of software testing and using any software development life cycle. In particular, we referred to the 2022 revision of the ISO/IEC/IEEE 29119 International Standard on Software and Systems Engineering — Software Testing, part 1 [21], which provides a general introduction to software testing, the role of software testing in V&V processes, how testing can be implemented, the concepts of test plan and test strategies, including test levels, test types and test design techniques.

B. Target Population

We decided to systematically search for academic courses teaching software testing topics in each of the European countries considered. Unlike the case of Sweden analysed by Barrett et al. [16], we could not find integrated national repositories of offered courses for any of the analysed countries, so we adopted the search approach described below.

In order to have a first list of the universities that offer computer science subjects in their degrees, we took into account the Scimago Institutions Ranking¹, which ranks academic

¹Scimago Institutions Ranking, <https://www.scimagoir.com/rankings.php>

TABLE I
NUMBER OF UNIVERSITIES AND COURSES INCLUDED IN THE ANALYSIS

	Spain	Italy	Belgium	Portugal	Overall
Universities in SJR ranking	62	70	10	29	171
Randomly Selected Universities	19 (31%)	20 (29%)	3 (30%)	7 (24%)	49 (29%)
Analysed Courses	28	44	10	35	117

institutions based on their research performance, innovation output, and societal impact as measured by their web visibility, allowing queries for specific countries, years and scientific fields. We focused on the 2023 rankings² of Computer Science Universities. The rankings included 62 institutions from Spain, 70 from Italy, 10 from Belgium and 29 from Portugal, for overall 171 universities.

We decided to construct our population by selecting a random sample equal to approximately 30% of all the 171 listed universities. We therefore selected at random 49 out of 171 universities (equal to 29%).

For each university, we examined the bachelor’s and master’s degrees offered through their institutional websites to identify courses related to software testing. As a priority, we analysed bachelor’s and master’s degrees related to computer science (i.e. Computer Engineering, Computer Science, Software Engineering, etc.). Therefore, we manually scanned the courses offered, looking for names that could be related to testing such as “Software Testing”, “Verification & Validation”, “Software Quality”, etc. We also considered “Software Engineering” and “Programming” courses, as they may typically deal with the basics of testing.

In order to filter only relevant courses, we applied the following inclusion criteria:

- The course syllabus contains a description of the course topics.
- The course syllabus is written in English or in the language of one of the ENACTEST Project partners, namely: Spanish, Italian, Portuguese and Dutch.
- The course syllabus includes testing topics.

In addition, we applied the following exclusion criterion:

- Cancelled courses (not active in 2022 and 2023).

At the end of this selection process, we found 117 courses that satisfied the inclusion and exclusion criteria and admitted them to the successive steps.

The number of universities listed in the Scimago Ranking and the number of universities we analysed for each country are shown in Table I. The Table also reports the number of analysed courses for each country, which were 28 from Spain, 44 from Italy, 10 from Belgium and 35 from Portugal.

C. Data Collection

To facilitate the collection of course information from each country, researchers who understand the language of each

²<https://www.scimagoir.com/rankings.php?sector=Higher+educ.\&area=1700&ranking=Overall>

country searched for the courses of the ranked universities and completed a data extraction form. We made sure to have several researchers per language. In this way, we ensured the correct identification of the courses and the validity of the information obtained.

The researchers studied the general characteristics of the courses by analysing the publicly available information, the syllabus and the curricula offered on the official websites of their academic institutions. The data extraction form we used to collect course information included the following fields:

- Country
- Name of the university
- Name of the degree
- Degree Level (e.g. Bachelor or Master)
- Course Name
- Course Year
- Teacher Name
- Number of course credits (EC)
- Number of hours (distinguished between Theory and Lab hours, when the information is available)
- Student Assessment Methods
- Course Syllabus
- Focus on Software Testing (Complete or Partial)
- List of testing topics included in the course
- Reference books

The focus of a Software Testing course can be either Complete or Partial, depending on the amount of software testing topics offered in the course. We classified courses as Software Testing courses (ST) if they had a majority focus on testing topics (i.e. more than 75% of topics correspond to testing), and as Non-Software Testing courses (NST) if software testing topics were only a minority of the course topics. This classification was finalised by reading the course syllabi. This process was carried out by two researchers who analysed all the courses and information and classified them as ST or NST.

With regard to the testing topics offered, we mapped them against the conceptual framework for testing offered by the ISO/IEC/IEEE 29119 standard on Software Testing. We focused on the dynamic testing approaches according to the standard, distinguishing: Test Design Techniques, Test Practices, Testing Levels, and Testing Types (see Table II).

Finally, two other researchers analysed the classifications collected from the courses to report the results.

The collected information are available online at <https://doi.org/10.5281/zenodo.10467218>

IV. RESULTS

A. RQ1: How common are software testing related courses in the considered academic context?

Overall we found 117 courses that include software testing topics in the 49 considered universities: we classified 22 of them as Software Testing (ST) courses and the remaining 95 ones as courses including some software testing topics (NST). Table III reports the number of mapped courses for each country.

TABLE II
TESTING APPROACHES ACCORDING TO THE ISO/IEC/IEEE 29119
STANDARD ON SOFTWARE TESTING

Test Design Technique	Testing Practice	Testing Type
Specification Based	Model-based testing	Functional testing
Equivalence Partitioning	Scripted testing	Accessibility testing
Classification tree method	Exploratory testing	Compatibility testing
Boundary value analysis	Experience-based testing	Conversion testing
Syntax testing	Manual testing	Disaster recover testing
Combinatorial testing	A/B testing	Installability testing
Decision table testing	Back-to-back testing	Interoperability testing
Cause-effect graphing	Mathematical-based testing	Localization testing
State transition testing	Fuzz testing	Maintainability testing
Scenario testing	Keyword-driven testing	Performance related testing
Use case testing	Automated testing	Portability testing
Random testing	Other	Procedure testing
Metamorphic testing		Reliability testing
Requirements-based testing		Security testing
Structure Based	Testing Level	Usability testing
Statement testing	Unit testing	Other
Branch testing	Integration testing	
Decision testing	System testing	
Branch condition testing	System integration testing	
Branch cond. comb. testing	Acceptance testing	
MC/DC testing	Other	
Data flow testing		
Experience Based		
Error guessing		
Other		

TABLE III
NUMBER OF ST AND NST COURSES FOUND IN THE UNIVERSITIES
SELECTED OF THE CONSIDERED COUNTRIES

	#Universities	#ST Courses	#NST Courses
Spain	19	7	21
Italy	20	5	39
Portugal	7	9	26
Belgium	3	1	9
Total	49	22	95

If we consider the ST courses and their diffusion in our population, 19 out of the 49 universities offered at least one of them. Only the Universities of Lisbon, Aveiro, and Coimbra, in Portugal, offered more than one distinct ST course, but in different educational programs. On average, we found ST courses in 39% of all the mapped universities. Regarding the 95 NST courses, 46 out of 49 universities offered at least one course (94%) that addresses in part software testing. If we consider both types of courses, 14 out of 49 universities (29%) offer both ST and NST courses.

Regarding the diffusion of ST courses in the different countries, 7 of them were present in 19 Spanish universities, 5 courses in 21 Italian ones, 9 in 7 Portuguese universities, and one course in one of the 3 mapped Belgian universities. If we consider the percentage of ST courses with respect to the number of analysed universities per country, we have ST courses in 37% of Spanish universities, 24% of Italian ones, 33% of Belgian ones, and 71% of Portuguese universities. By comparing these results with those reported in the literature, we find that the frequency of ST courses in Spain, Belgium and Italy is lower than in Sweden, where Barrett et al. [16] recently found 14 ST courses in 25 different universities (56%). Only the frequency of ST courses in Portugal was 71% and larger than in Sweden.

TABLE IV
ST COURSE- OVERVIEW CHARACTERISTICS

Co.	University	Course Name	Deg.	Yr.
BE	Antwerpen	Software testing	MSc	1
ES	Barcelona (Autonoma)	Test and software Quality	BSc	3
ES	Madrid (Complutense)	Software Testing	BSc	3
ES	Madrid (Politécnica)	Software validation and verification	MSc	1
ES	Oviedo	Software Quality, Validation and Verification	BSc	4
ES	Alacant	Planning and Testing of Software Systems	BSc	3
ES	Valencia (Politecnica)	Software testing	MSc	2
ES	Zaragoza	Validation and Verification	BSc	3
IT	Bergamo	Testing e Verifica del Software	MSc	2
IT	Firenze	Advanced Programming Techniques	MSc	1
IT	Milano	Verifica e Convalida del Software	MSc	1
IT	Milano (Bicocca)	Software Quality	MSc	1
IT	Napoli (Federico II)	Software Testing	MSc	1
PT	Aveiro	Robust Software	MSc	1
PT	Aveiro	Software Testing and Quality Control	BSc	3
PT	Aveiro	Software Testing	MSc	1
PT	Coimbra	Analysis of Software Artifacts	MSc	1
PT	Coimbra	Software Quality and Dependability	MSc	1
PT	Lisbon	Software Verification and Validation	MSc	1
PT	Lisbon	Software Testing and Validation	MSc	1
PT	Minho	Testing and Validation of Information Systems	MSc	1
PT	Porto	Software Testing, Verification and Validation	MSc	2

B. RQ2 : What are the educational organisational properties of the courses?

Tables IV and V provide an overview on the characteristics of respectively the ST and NST courses we mapped. In this section we analyse the main characteristics of the courses, including their Educational Level, Curriculum, Year, Course Names, Number of Credits and Assessment Methods.

1) Educational Level, Curriculum and Year of courses:

In terms of educational level, 6 ST courses are offered at the bachelor level (always in the final year) and 16 at the master level. In terms of curriculum, 15 courses are offered by Computer Science curricula, whereas the remaining 7 ones belong to Computer Engineering ones.

66 out of 95 NST courses are at the bachelor level, while the remaining 29 courses are at the master level. As to the educational programs, 47 belong to Computer Science curricula, 46 are in Computer Engineering curricula, and the two remaining ones are offered in other scientific fields.

It is interesting to note that NST courses are equally offered in the context of Computer Science and Computer Engineering curricula, while ST courses are more common in Computer Science than in Computer Engineering ones. A possible explanation of this datum is that CS curricula are typically more focused on software development topics than CE curricula that usually have to give space also to non-software related topics from other engineering areas.

Figure 1 illustrates the current offering of ST and NST courses from our population, by distinguishing them on the basis of the corresponding Educational Level (Master or Bachelor) and Year. As the Figure shows, NST courses are most often offered in the third year of bachelor's degrees or in the first year of master's degrees. ST courses, on the other hand, are always offered starting from the third year of bachelor's degrees and are mostly present in the first year of master's degrees.

2) Course Names : We analysed the courses in order to find the most frequent terms included in their names. For the ST courses, the most common terms were: "Software Testing" (10

TABLE V
NST COURSE- OVERVIEW CHARACTERISTICS

Co.	University	Course Name	Deg.	Yr.
BE	Antwerpen	Project software engineering	BSc	1
BE	Antwerpen	Software engineering	BSc	3
BE	Gent	Software Development & Operations	BSc	3
BE	Leuven (KUL)	Digital Design Concepts	BSc	2
BE	Leuven (KUL)	Object-gericht programmeren	BSc	1
BE	Leuven (KUL)	Objectgerichte softwareontwikkeling	BSc	2
BE	Leuven (KUL)	Programmeertechnieken	BSc	2
BE	Leuven (KUL)	Software engineering en webtechnologie	BSc	3
BE	Leuven (KUL)	Software-ontwerp	BSc	3
ES	Alcalá	Software Engineering	BSc	1
ES	Barcelona (Autonoma)	Software Engineering	BSc	2
ES	Barcelona (Politecnica)	Software Architecture	BSc	3
ES	Barcelona (Politecnica)	Software Architecture	BSc	3
ES	Barcelona (Politecnica)	Software Architecture	BSc	3
ES	Castilla La Mancha	Software engineering II	BSc	3
ES	Castilla La Mancha	Software engineering II	BSc	3
ES	Granada	Software Development	BSc	3
ES	Madrid (Autónoma)	Software Engineering	BSc	3
ES	Madrid (Carlos III)	Software Development	BSc	2
ES	Madrid (Rey Juan Carlos)	Software Quality	BSc	3
ES	Málaga	Introduction to Software Engineering	BSc	1
ES	Murcia	Software Quality	BSc	4
ES	Oviedo	Quality of Product and Processes	MSc	1
ES	Pais Vasco	Software Engineering II	BSc	3
ES	Sevilla	Design and Testing I	BSc	3
ES	Sevilla	Design and Testing II	BSc	3
ES	Valencia	Software production methods	MSc	1
ES	Valencia (Politecnica)	Audit, Quality and Management of Information Systems	MSc	1
ES	Valencia (Politecnica)	Audit, Quality and Management of Information Systems	MSc	1
ES	Zaragoza	Software Engineering	BSc	3
IT	Benevento	Software Engineering	BSc	3
IT	Bergamo	Software Engineering	BSc	3
IT	Brescia	Software Engineering	BSc	3
IT	Campobasso	Software Engineering	BSc	3
IT	Firenze	Programming Methodologies	BSc	2
IT	Genova	Advanced Programming Techniques	BSc	3
IT	Genova	Functional and Security Testing Techniques	MSc	1
IT	Genova	Software Engineering	MSc	1
IT	Genova	Software Engineering Fundamentals	BSc	3
IT	Milano (Bicocca)	Software Analysis and Design	BSc	2
IT	Milano (Politecnico)	Software Engineering	BSc	3
IT	Milano (Politecnico)	Software Engineering	BSc	3
IT	Milano (Politecnico)	Software Engineering	BSc	3
IT	Milano (Politecnico)	Software Engineering	BSc	3
IT	Milano (Politecnico)	Software Engineering 2	MSc	1
IT	Milano (Politecnico)	Software Engineering 2	MSc	1
IT	Milano (Politecnico)	Software Engineering 2	MSc	1
IT	Modena	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Engineering	BSc	3
IT	Napoli (Federico II)	Software Project Management and Evolution	MSc	1
IT	Napoli (Unicampania)	Software Engineering	BSc	3
IT	Padova	Software Engineering	BSc	3
IT	Pisa	Software Engineering	BSc	3
IT	Roma (Sapienza)	Software Design	BSc	2
IT	Salerno	Ingegneria, Gestione ed Evoluzione del Software	BSc	3
IT	Salerno	Management and Evolution of Software Projects	MSc	1
IT	Salerno	Software Dependability	MSc	2
IT	Salerno	Software Engineering	BSc	3
IT	Torino	Istituzioni di Sviluppo Software	MSc	1
IT	Torino (Politecnico)	Object Oriented Programming	BSc	2
IT	Torino (Politecnico)	Object Oriented Programming	BSc	2
IT	Torino (Politecnico)	Object Oriented Programming	BSc	2
IT	Torino (Politecnico)	Software Engineering	MSc	1
IT	Torino (Politecnico)	Software Engineering	MSc	1
IT	Torino (Politecnico)	Software Engineering	BSc	2
IT	Verona	Software Engineering Fundamentals	MSc	1
PT	Aveiro	Analysis and Exploration of Vulnerabilities	MSc	1
PT	Aveiro	Digital Accessibility and Compliance	MSc	1
PT	Aveiro	Software Engineering	BSc	2
PT	Aveiro	Usability and User Experience	MSc	1
PT	Coimbra	Design and Development of Secure Software	MSc	1
PT	Lisbon	Software Engineering	BSc	3
PT	Lisbon	Programming labs	BSc	1
PT	Lisbon (ISCTE)	Agile Software Development	BSc	2
PT	Lisbon (ISCTE)	Software Engineering	BSc	3
PT	Lisbon (Nova)	Object Oriented Programming	BSc	1
PT	Lisbon (Nova)	Software Construction and Verification	MSc	1
PT	Lisbon (Nova)	Software Engineering	BSc	3
PT	Lisbon (Nova)	Software Quality	MSc	1
PT	Minho	Cybersecurity	MSc	1
PT	Minho	Informatics Laboratories I	BSc	1
PT	Minho	Information Systems Security Engineering	MSc	1
PT	Minho	Object Oriented Programming	BSc	2
PT	Minho	Security Engineering	MSc	1
PT	Minho	Security Technologies	MSc	1
PT	Minho	Software Systems Development	BSc	3
PT	Porto	Programming	BSc	1
PT	Porto	Software Design and Testing Laboratory	BSc	2
PT	Porto	Software Engineering	BSc	2
PT	Porto	Secure Software Engineering	MSc	2
PT	Porto	Security in Software Engineering	MSc	1
PT	Porto	Software Engineering Laboratory	MSc	1

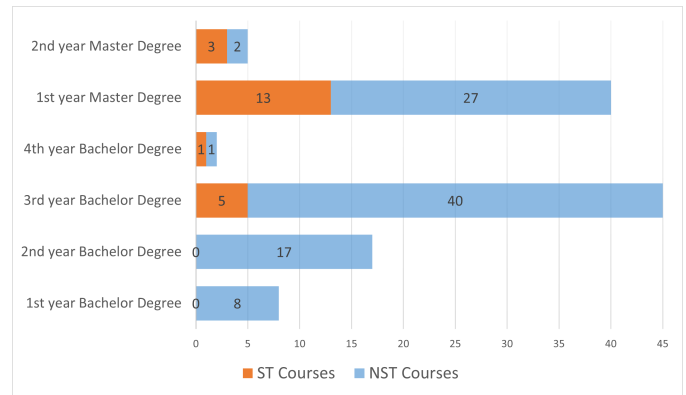


Fig. 1. Courses by Educational Level and Year

times), “Verification and Validation” (8 times), and “Software Quality” (5 times). In a few cases, the name of the course may include less typical terms, such as in courses named “Analysis of Software Artifacts”, “Planning and Testing of Software Systems”, “Robust Software”, and “Advanced Programming Techniques”.

Regarding the NST course names, the most frequent terms were “Software Engineering” (44 times), “Programming” (20 times), “Software Architecture Design” (10), “Security” (9), “Software Quality” (7), “Software Project Management” (6), and “Agile Software Development” (1). These results confirm that software testing topics are usually introduced in Software Engineering courses, but also in many Programming, Software Architecture design, Security, and Quality courses.

3) *Number of Course Credits (EC)*: We analyzed the number of European Credits (EC) associated with each course, EC being a standard system for measuring the effort required by a student for that course. We also tried to analyse the number of hours of each course, distinguishing between the theoretical, practical, and laboratory hours. Unfortunately, this information could not be extracted from the websites of each course analysed, as it was not always available or no standard description according to these categories was provided.

In terms of the number of ECs, almost all of the 22 mapped ST courses correspond to 6 ECs, with only three exceptions of 5, 4 and 2 ECs respectively. The average EC value of ST courses is 5.68 credits.

We made a similar analysis of the NST courses. They have EC values ranging from 3 to 12 ECs, with an average of 6.75 credits. However, it was not possible to determine exactly how much of the course time was spent on testing.

4) *Assessment Methods*: Regarding the assessment, we collected the different approaches used by the courses and abstracted 6 types of assessment methods. We classified the methods into: Open questions, Closed questions, Exercises, Homework, Project, and Discussion. Open and Closed questions assessments are based on a questionnaire. Exercises assessments require students to complete practical exercises. Homework assessments are based on the evaluation of homework assigned to students. Project means that the assessment is

based on a project usually carried out by students organized in teams. Discussion represents a more traditional method based on a talk or presentation on topics provided by the course.

Figure 2 summarizes the types of assessment methods and the percentages of ST and NST courses adopting them. As the Figure shows, ST and NST courses present similar percentages for each method.

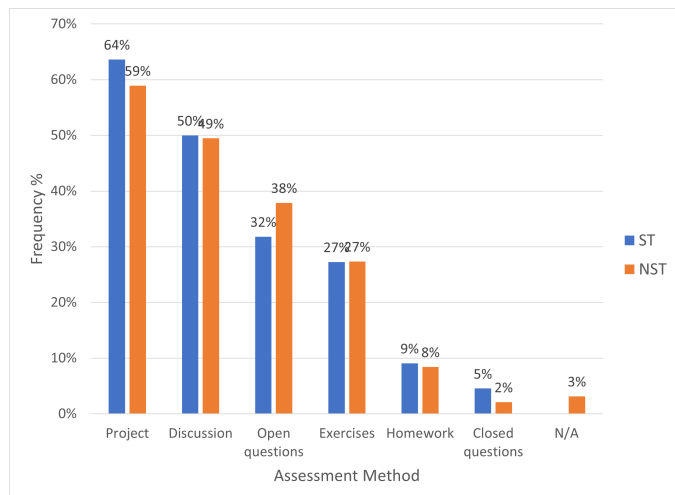


Fig. 2. Frequency of Assessment Methods in Mapped ST and NST Courses

Regarding ST course assessments, the majority of courses (14 out of 22, 64%) rely on a Project, whereas a Discussion is used in 11 out of 22 courses (50%). The use of Open questions or Exercises (including either written exercises or lab exercises) is quite frequent (respectively in 32% and 27% of courses), whereas Homework or Closed questions are less common (respectively in 9% and 5% of courses).

Analogously for NST courses, the most common assessment method is the Project (in 59% of courses) followed by the Discussion (49%), Open Questions (38%), written or lab Exercises (27%), Homework (8%) and Closed Questions (only in 2% of courses). For 3 NST courses (3%) we could not find detailed information about their assessment methods and reported them as N/A.

The data we collected from the publicly available course descriptions also shows that most of the mapped courses adopt more than one assessment method. On average, ST and NST courses adopt 1.86 and 1.87 methods, respectively. Both assessment methods that are suitable for assessing the knowledge of theoretical topics (like Open/Closed Questions and Discussion) and other methods more suitable for assessing practical skills (like Projects, Homework, and Exercises) are typically used. This result confirms the dual nature (both Theoretical and Practical) of most of the courses analysed.

C. RQ3: What aspects of software testing are most commonly taught?

To answer this question, we reviewed the contents of the syllabi available on the Web pages describing each course. Our aim was to map the topics taught with respect to the

categories of Test Design Techniques, Test Practices, Testing Levels, and Testing Types proposed by the ISO/IEC/IEEE 29119-1 Standard on Software Testing, that we reported in Table II.

Since not all course syllabi conform to the Standard terminology and many provide only high-level lists of test topics, the involved authors had to investigate further to abstract the topics covered, and sometimes they had to make some interpretations. In the latter cases, a third author double-checked the interpretations made by two authors to exclude possible classification errors. For example, the terms "black box testing" and "white box testing" were often found. In these cases, terms were interpreted as *Specification Based* and *Structure Based* test design techniques, respectively. In other cases, only the term "statement testing" was found and it was assumed that the *Structure Based* category was also covered in the course.

Table VI shows the number and percentage of ST courses and NST courses in which each testing topic is covered.

1) *Test Design Techniques*: With respect to the three main Standard categories of Test Design Techniques (e.g., Specification Based, Structure Based, and Experience Based), we found that they are included in 95%, 100%, and 9% of ST course topics, respectively, indicating a slight prevalence of Structure Based compared to Specification Based. As to the Specification Based techniques, the most frequently taught topics are Equivalence Partitioning (in 50% of ST courses), followed by Boundary Value analysis (27%), Decision Table testing (27%), Combinatorial testing (27%) and Random Testing (18%). The Syntax testing technique is taught in 3 courses (14%) while Cause-effect graphing and Metamorphic testing are both mentioned in only 2 courses (9%). Surprisingly, the topics of Use Case testing, Scenario testing, and Requirements-based testing are only explicitly mentioned in one course each, but we could not deduce whether they are actually so uncommon, or they are rather described in the syllabi as belonging to the more general category of Specification Based techniques.

Regarding the Structure Based techniques, the most frequently taught are Statement testing (23%), Branch testing (23%), Decision testing (18%), Branch Condition (18%), Branch condition combination (18%) and MC/DC testing (14%). Data-flow testing is also quite common, being taught in 23% of ST courses.

As to the category of Experience Based, we encountered it in only 2 ST courses (9%), while Error Guessing is never mentioned.

Regarding testing techniques not explicitly mentioned in the standard, we found 4 courses where mutation-based testing techniques are taught and reported these in the category Other of the Table (4 out of 22, 18% of courses).

Regarding the NST courses, since testing is not the main focus of these courses, we often found in the syllabi only references to generic test design techniques, like Specification Based, found in 35 courses (37%), or Structure Based techniques found in 29 (31%) of courses.

TABLE VI
TESTING TOPICS OCCURRENCE IN COURSE SYLLABI

		ST	%	NST	%
Test	Specification Based	21	95%	35	37%
Design Technique	Equivalence Partitioning	11	50%	1	1%
	Classification tree method	1	5%	0	0%
	Boundary value analysis	6	27%	2	2%
	Syntax testing	3	14%	0	0%
	Combinatorial testing	6	27%	2	2%
	Decision table testing	6	27%	0	0%
	Cause-effect graphing	2	9%	0	0%
	State transition testing	8	36%	0	0%
	Scenario testing	1	5%	0	0%
	Use case testing	1	5%	0	0%
	Random testing	4	18%	0	0%
	Metamorphic testing	2	9%	0	0%
	Requirements-based testing	1	5%	3	3%
	Structure Based	22	100%	29	31%
	Statement testing	5	23%	9	9%
	Branch testing	5	23%	6	6%
	Decision testing	4	18%	7	7%
	Branch condition testing	4	18%	2	2%
	Branch condition combination testing	4	18%	0	0%
	MC/DC testing	3	14%	1	1%
Data flow testing	5	23%	0	0%	
Experience Based	2	9%	0	0%	
Error guessing	0	0%	0	0%	
Other	4	18%	12	13%	
Testing Practice	Model-based testing	7	32%	5	5%
	Scripted testing	3	14%	6	6%
	Exploratory testing	3	14%	1	1%
	Experience-based testing	1	5%	0	0%
	Manual testing	3	14%	6	6%
	A/B testing	0	0%	4	4%
	Back-to-back testing	0	0%	0	0%
	Mathematical-based testing	3	14%	0	0%
	Fuzz testing	2	9%	0	0%
	Keyword-driven testing	0	0%	0	0%
	Automated testing	15	68%	20	21%
	Other	6	27%	26	27%
	Testing Level	Unit testing	13	59%	48
Integration testing		10	45%	22	23%
System testing		10	45%	18	19%
System integration testing		4	18%	0	0%
Acceptance testing		7	32%	12	13%
Other	4	18%	12	13%	
Testing Type	Functional testing	8	36%	40	42%
	Accessibility testing	0	0%	1	1%
	Compatibility testing	0	0%	0	0%
	Conversion testing	0	0%	0	0%
	Disaster recover testing	0	0%	0	0%
	Installability testing	0	0%	0	0%
	Interoperability testing	0	0%	0	0%
	Localization testing	0	0%	0	0%
	Maintainability testing	0	0%	0	0%
	Performance related testing	4	18%	6	6%
	Portability testing	0	0%	0	0%
	Procedure testing	0	0%	0	0%
	Reliability testing	0	0%	2	2%
	Security testing	2	9%	9	9%
	Usability testing	1	5%	1	1%
	Other	5	23%	9	9%

2) *Testing Practice*: Automated Testing is the most common testing practice included in the syllabi of both types of mapped courses. It is explicitly mentioned in 15 ST courses (68%) and 20 NST courses (21%). This datum indicates that automated testing is commonly taught in specialized courses of testing but less frequently in non-specialized ones.

The second most common practice is Model-based testing, which was found in 7 ST courses (32%) but only in 5 NST courses (5%). Conversely, Scripted testing is taught in 3 ST courses but also in 6 NST ones.

Other testing practices that are less frequently taught in ST courses include Exploratory testing, Manual testing, Mathematical-based testing (each of them in 3 ST courses),

Fuzz testing (in 2 courses) and Experience-based testing (only in one ST course). Regarding NST courses, except for 6 courses mentioning Manual testing and a single course mentioning Exploratory testing, none of the latter practices was found in the mapped courses.

Overall, we found no mentions at all of Back-to-back and Keyword-driven testing, while A/B testing is included in the syllabus of only 4 NST courses.

We also observed mentions of 'Other' common testing practices, such as Regression Testing (in a total of 10 courses) and Test Driven Development (in a total of 3 courses).

3) *Testing Level*: Regarding Testing Levels, there are different scenarios in ST and NST courses. In ST courses Unit testing, Integration testing and System testing are often included in the syllabi (respectively in 59%, 45% and 45% of the mapped ST courses), whereas in NST courses Unit testing (in 48 courses, 51%) is more frequent than Integration testing (in 22 courses, 23%) and System testing (18 courses, 18%). Analogously, Acceptance testing was more often found on average in ST courses (in 7 courses, 32%) than in NST courses (12 courses, 13%). Finally, System Integration testing was found only in 4 ST courses (18%).

4) *Testing Type*: Finally, regarding testing types, we found that Functional testing was by far the most cited type of testing in course syllabi: we found it in 8 ST courses (36%) and 40 NST courses (42%). According to the Standard [21], functional testing is used to check the implementation of functional requirements, while non-functional types of testing are used to check that requirements in other areas, like performance and security, are met.

However, based on the syllabus reviews and our own experience in software testing, we can assume that many ST and NST courses are predominantly based on functional testing rather than non-functional testing. As a consequence, the reference to this type of testing by all these courses should be reported, even if it is not explicitly declared.

The other 3 testing types that were sometimes found in the mapped courses included Performance related testing (found in 4 ST and 6 NST courses), Security testing (found in 2 ST courses and 9 NST courses, mostly devoted to secure software design and development) and Usability testing (only in one ST course and in one NST course). Accessibility testing is mentioned in a single NST course only.

The remaining types of testing proposed by the standard, including Compatibility, Conversion, Disaster Recovery, Installability, Interoperability, Localization testing, Maintainability, Portability, Procedural, and Reliability testing, were not mentioned at all.

D. Threats to validity

Although we planned and executed the mapping carefully, some threats could affect the validity of the results [22], [23].

Construct Validity. The lack of a standard language and terminology in the syllabi descriptions found on the Web could affect the construct validity of the course mapping. To mitigate this threat, we decided that two researchers who perfectly

understand the language of the course will gather the information to ensure that the information is collected correctly without misinterpretation, and we used the ISO/IEC/IEEE 29119 Standard to classify the subjects taught.

In addition, we are aware that educators may interpret some terms reported in the syllabi in different ways, which could threaten our study. For example, our collected data on hours of theory and practice uses directly the information provided online, without interpretation. Therefore, there is a risk that different educators consider the terms differently. To mitigate this threat, we intend to survey or interview teachers in future work.

We also recognise that a possible threat is that the information publicly available may not fully reflect the contents of the courses. Regardless, these still provide a good indication of the structure of each course and, as these are provided to potential applicants, we assume that the information is most likely valid and trusted.

Internal Validity. In order to minimize the threats associated with processing the available course information, we decided to only work with the public information available on the websites. While all of the courses had basic information available online and, therefore, none of the selected universities' programmes or courses had to be removed for the lack of information, the level of detail provided varied. In some courses, the information was abstract, making it more challenging to obtain detailed information about the testing topics covered in the course. To ensure the accuracy of the interpretation of the online information, all the entries were checked by at least two researchers and an extra researcher in case of doubts. In addition, we make all the data collected available to the other researchers to verify the validity of the results.

External Validity. To avoid bias in the selection of universities, we randomly selected those ranked in Scimago, which comprises universities that often have well-established teaching practices and resources, so our study still reflects trends and standards in teaching software testing that are likely to be observed in other higher education institutions.

The courses we selected may not be generalized to other countries, nor may they provide an accurate reflection of the current general landscape of software testing education in Europe at large. To mitigate this threat, we defined a protocol for the selection of the universities and courses, and reported the criteria and methods used for course selection, so that these can be used by other researchers to replicate the study and increase the body of knowledge of the state of the practice of software testing teaching.

V. DISCUSSION

As to the diffusion of software testing courses investigated in RQ1, the study results show that courses specifically focused on software testing are available at only 39% of the universities surveyed and are mostly offered at Master degree level. This percentage is slightly lower than the one reported by Ardic and Zaidman, who found software testing

dedicated courses in 50% of the top 100 of the Times Higher Education university ranking [15], as well as lower than the 56% of universities with dedicated ST courses in the Sweden context [16]. While it was noteworthy that software testing fundamentals were included in at least one course at every university (i.e., in 94% of surveyed universities), most of the existing software testing techniques are not taught to students and future IT professionals. This is far too limited considering that we live in a world surrounded by software whose quality could potentially disrupt our lives.

As to the pedagogical approaches investigated by RQ2, as we stated before, we did not find the number of hours of theoretical classes and practical classes for each mapped course. Nevertheless, for those for which we found this information (59% of ST courses), we observe that they spend similar amount of hours for theoretical and practical classes, with a ratio of 1.03 between them. As it has been discussed in [24], the emphasis in teaching testing should be placed primarily on the practical part. This approach is both a more challenging and attractive strategy for students, and also helps students to develop the ability to apply testing concepts in real-world development scenarios.

The results present interesting insights about assessment methods in both ST and NST courses. Projects are used as the assessment method in almost 60% of courses. This is aligned with the practical experience needed to learn complex topics, since during the development of projects students can practice the application of concepts.

Regarding the information on reference books, we cannot get a clear list of software testing books used. If we look at the NST courses, they mainly use general software engineering books. On the other hand, if we look at the ST courses, not all of them provide information about the books, and those that do, use a large variety of books related to the topics taught.

As regards the software testing topics that are most commonly taught (RQ3), specification-based and structure-based Test Design techniques included in the ISO Standard are taught in almost all the analysed courses, whereas the experience-based category is present in only 2 of the 117 considered courses. Analogously, as to the Testing Practices listed by the Standard, the results of the mapping study show that only a small percentage (3,41%) of courses teach exploratory testing (4 out of 117 courses). Our analysis reflects that testing courses of universities are more inclined towards the *analytical school*, where the emphasis is on better testing through improved precision of specifications in stead of the *context-driven school* that emphasizes exploratory testing, which promotes concurrent learning, test design, and test execution [25].

A final consideration concerns the Testing Types that are commonly taught. Our study revealed areas of deficiency that require attention to bring current academic offerings closer to the real needs of the industry. Some clear findings include the lack of accessibility testing, security testing, and disaster recovery testing, among others.

Our study can help academic institutions to understand the gaps in the curriculum, and consider some ways to close

these gaps by paying attention to the underrepresented testing topics and techniques. In addition, sharing this knowledge can encourage collaborations between different educational institutions.

VI. CONCLUSIONS

This paper presents a landscape of software testing teaching practices in four Western European countries, based on the mapping of 117 courses syllabi within 49 Universities.

Our analysis is based on the course descriptions provided in the summer and autumn of 2023. The content of courses is typically adjusted as time evolves. Therefore, the results have to be considered as a snapshot of the situation in 2023.

With regard to related work (SLRs on software testing education and mapping of software testing courses), our study provides an updated overview of courses in four European countries and contributes with a clear characterisation of the academic properties of courses. To the best of our knowledge, this is the first study based on the mapping of more than 100 courses and providing a fine-grained characterisation of testing topics based on the 2021 revision of the ISO/IEC/IEEE 29119 International Standard on Software Testing.

A difficulty we have found in analyzing the academic characteristics of courses is that the information is not provided in a standardised way, nor do we have a consolidated repository of courses in Europe. This challenged us to define a protocol to search the information, starting with the partner countries from the ENACTEST Project. Similarly to the Bologna Process³ the European Credit Transfer and Accumulation System (ECTS) that was adopted across the continent some years ago, a standard way of defining the academic characteristics of courses should be systematically adopted to facilitate the analysis. We consider this as an important future direction for educational contexts.

In future work we intend to survey the teachers of the courses to gather further information that did not emerge from the analysis of the syllabi and to investigate the reasons for the choices they made.

We also found that most courses focus exclusively on testing according to the analytical school of thought. This is overly restrictive and one-sided, especially in today's rapidly evolving technology landscape. With the proliferation of AI systems, it is imperative that contextual approaches be incorporated even more prominently. The emergence of AI introduces a set of quality characteristics that go beyond traditional, clear-cut specifications. Attributes such as intelligence, accountability, and explainability are integral to AI testing and can only be adequately assessed through context-based exploratory testing. As AI technologies continue to shape our world, it will become even more important for testing methodologies to adapt and encompass these complex, context-dependent facets to ensure that software testing education remains aligned with the evolving needs of the industry.

³European Higher Education Area and Bologna Process, <https://www.ehea.info/index.php>

The key takeaways from this study are:

- The systematic adoption of a standardized approach for describing the academic characteristics of courses is essential to streamline the analysis process of testing courses and the reference books used.
- The use of projects is the most prevalent method of assessing knowledge of testing practices, likely because of its effectiveness.
- Inclusion of exploratory testing topics in course content is necessary, which includes the context of the systems instead of just emphasizing the precision of testing specifications.
- The evaluation of software testing topics may reveal shortcomings that require attention to better align current academic offerings with real industry needs that we are currently investigating in the context of ENACTEST Project.
- Increasing the academic offering of specialized software testing courses may be necessary to improve the overall preparation of students.

Immediate future work will consider the creation of capsules to improve testing education in a seamless way across courses of computer science curricula. These capsules will take into account the less taught testing topics. We also plan to conduct empirical studies to uncover students' sensemaking during testing, as well as to develop a cognitive model of testing that may be useful for improving students' learning effectiveness.

ACKNOWLEDGMENT

This work has been partially funded by ENACTEST (European innovation alliance for testing education), ERASMUS+ Project number 101055874, 2022-2025 and by GATT (Gamification in Testing Teaching), funded by the University of Naples Federico II Research Funding Program (FRA).

REFERENCES

- [1] "Tricentis Software Fail Watch Finds 3.6 Billion People Affected and \$1.7 Trillion Revenue Lost by Software Failures Last Year," <https://www.globenewswire.com/news-release/2018/01/24/1304535/0/en/Tricentis-Software-Fail-Watch-Finds-3-6-Billion-People-Affected-and-1-7-Trillion-Revenue-Lost-by-Software-Failures-Last-Year.html>, 2018, accessed: 2024-01-18.
- [2] H. Krasner, "Cost of Poor Software Quality in the U.S.: A 2022 report," *Consortium for Information and Software Quality TM (CISQTM)*, pp. 1–61, 2022, accessed: 2024-01-18. [Online]. Available: <https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/>
- [3] —, "The cost of poor software quality in the US: A 2020 report," *Proc. Consortium Inf. Softw. QualityTM (CISQTM)*, pp. 1–46, 2021.
- [4] "CVE," <https://www.cve.org/>, accessed: 2024-01-18.
- [5] V. Garousi, M. Felderer, M. Kuhmann, K. Herkiloğlu, and S. Eldh, "Exploring the industry's challenges in software testing: An empirical study," *Journal of Software: Evolution and Process*, vol. 32, no. 8, 2020.
- [6] K. Juhnke, M. Tichy, and F. Houdek, "Challenges concerning test case specifications in automotive software testing: assessment of frequency and criticality," *Software Quality Journal*, vol. 29, pp. 39–100, 2021.
- [7] A. Afzal, C. Le Goues, M. Hilton, and C. S. Timperley, "A study on challenges of testing robotic systems," in *13th International Conference on Software Testing, Validation and Verification (ICST)*. IEEE, 2020, pp. 96–107.
- [8] T. Cowling, "Stages in teaching software testing," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 1185–1194.

- [9] L. P. Scatalon, J. C. Carver, R. E. Garcia, and E. F. Barbosa, "Software testing in introductory programming courses: A systematic mapping study," in *50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 421–427.
- [10] V. Garousi, A. Rainer, P. Lauvås Jr, and A. Arcuri, "Software-testing education: A systematic literature mapping," *Journal of Systems and Software*, vol. 165, 2020.
- [11] B. Marín, T. E. J. Vos, A. C. Paiva, A. R. Fasolino, and M. Snoeck, "ENACTEST-European Innovation Alliance for Testing Education." in *RCIS Workshops*, 2022. [Online]. Available: <https://ceur-ws.org/Vol-3144/RP-paper5.pdf>
- [12] B. Marín, T. E. J. Vos, M. Snoeck, A. C. R. Paiva, and A. R. Fasolino, "ENACTEST project - european innovation alliance for testing education," in *Research Projects Exhibition Papers Presented at the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023)*, ser. CEUR Workshop Proceedings, vol. 3413, 2023, pp. 91–96. [Online]. Available: <https://ceur-ws.org/Vol-3413/paper13.pdf>
- [13] V. Garousi and A. 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*, 2010, pp. 89–96.
- [14] P. H. D. Valle, E. F. Barbosa, and J. C. Maldonado, "Cs curricula of the most relevant universities in brazil and abroad: Perspective of software testing education," in *2015 International Symposium on Computers in Education (SIEE)*, 2015, pp. 62–68.
- [15] B. Ardic and A. Zaidman, "Hey teachers, teach those kids some software testing," in *5th International Workshop on Software Engineering Education for the Next Generation (SEENG)*. IEEE, 2023, pp. 9–16.
- [16] A. A. Barrett, E. P. Enoiu, and W. Afzal, "On the current state of academic software testing education in sweden," in *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2023, pp. 397–404.
- [17] "ENACTEST project," <https://enactest-project.eu>, last accessed: 2024-01-18.
- [18] I. S. Elgrably and S. R. B. de Oliveira, "A diagnosis on software testing education in the brazilian universities," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–8.
- [19] S. M. Melo, V. X. S. Moreira, L. N. Paschoal, and S. R. S. Souza, "Testing education: A survey on a global scale," in *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, ser. SBES '20. Association for Computing Machinery, 2020, p. 554–563. [Online]. Available: <https://doi.org/10.1145/3422392.3422483>
- [20] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994.
- [21] "Iso/iec/ieee international standard - software and systems engineering –software testing –part 1:general concepts," *ISO/IEC/IEEE 29119-1:2022(E)*, pp. 1–60, 2022.
- [22] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.
- [23] P. Ralph and E. Tempero, "Construct validity in software engineering research and software metrics," in *22nd International Conference on Evaluation and Assessment in Software Engineering*, 2018, pp. 13–23.
- [24] B. Marín, S. Alarcón, G. Giachetti, and M. Snoeck, "Tescav: An approach for learning model-based testing and coverage in practice," in *14th International Conference on Research Challenges in Information Science, RCIS*. Springer, 2020, pp. 302–317.
- [25] N. Doorn, T. E. Vos, and B. Marín, "Towards understanding students' sensemaking of test case design," *Data and Knowledge Engineering*, vol. 146, p. 102199, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0169023X23000599>