2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW) | 979-8-3503-3335-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICSTW58534.2023.00072

ISTQB-based Software Testing Education: Advantages and Challenges

1st Attila Szatmári
Software Engineering Department
University of Szeged
Szeged, Hungary
szatma@inf.u-szeged.hu

2nd Tamás Gergely
Software Engineering Department
University of Szeged
Szeged, Hungary
gertom@inf.u-szeged.hu

3rd Árpád Beszédes
Software Engineering Department
University of Szeged
Szeged, Hungary
beszedes@inf.u-szeged.hu

***Abstract*—The International Software Testing Qualifications Board (ISTQB) is a widely-recognized organization that provides a certification scheme in the field of software testing. The knowledge this scheme sets for different levels and areas of software testing has become a standard terminology and requirement for software testers in the industry. In the first part of this paper, in a small survey, we examine the importance of certification in the industry, then we do a literature review to investigate to what extent the ISTQB material is used in university curricula. We examined job-offering portals and discovered that LinkedIn and Jobline mostly had job listings that favored the certificate, however, many jobs required the same skills and knowledge that the ISTQB Foundation Level certificate provides. Therefore, we recommend that universities include software testing in their computer science curriculum and accommodate the principles outlined by ISTQB. In the last part of the paper, we highlight the challenges of using ISTQB material in education based on our experience and show some possible solutions to overcome them. In conclusion, by providing sufficient knowledge, universities can provide an opportunity for students and companies to save time spent on post-graduate training for test-related positions.**

***Index Terms*—ISTQB, Test Education, Higher Level Education, Testing**

I. Introduction

Software Testing is a vital step in the software development process. A badly written software program can cause problems of all sorts, from minor inconveniences to even potentially life-threatening situations. Skilled software testers are able to find these problems before the software is released to the users. Therefore, it is important to teach computer science (CS) students how to test software effectively [1]–[3]. Gariousi et al. [4] show that requirements, design, and testing are the most important skills in the industry. Hence, it is necessary to investigate effective approaches for teaching students about testing in academic environments. Even if CS graduates do not pursue careers as testers, they can identify problems easier before testers even see them [5]. This practice makes software programs more robust in general.

ISTQB®[1], short for International Software Testing Qualifications Board, is a non-profit organization that offers educational resources and certifications for testers on a global scale. These are based on widely accepted testing principles and practices used in the industry. Those who take the path of becoming software testers face their first challenge in their job application interviews. Interviewers usually ask as a bonus question about whether the applicant has any tester certifications like ISTQB CTFL (Certified Tester Foundation Level) or a higher level ISTQB certification. Usually, companies prefer the job applicant to have some experience, but it is even better if they have at least a foundation-level ISTQB certificate. Nowadays, this became the defacto entry-level knowledge for the majority of software tester jobs.

While ISTQB provides excellent materials for software testers, there are plenty of other sources, standards, and books from which testers can extend their knowledge. One example is the international standard (ISO/IEC/IEEE 29119 standard family [6]–[9]), which ISTQB also acknowledges and refers to further details on the subjects they cover. Additionally, the Art of Software Testing by Myers et al. [10] provides a brief but comprehensive presentation on software testing approaches, and also gives examples of "Absence-of-errors is a fallacy" testing principles alongside Kaner [11], Weinberg [12], Jorgensen [13], and Copeland [14].

Computer Science students often take courses in software development that cover a variety of programming languages and software testing to some extent [15]–[17]. However, few of these courses focus only on testing. As a result, students may only learn the basics and the technical aspects of unit testing and may not be able to effectively ensure the quality of their software. This highlights the importance of including testing as a key part of a Computer Science education. By learning how to test software effectively, students can better ensure the quality and reliability of the software they develop. Some Hungarian universities have testing courses in their curricula, which are based on testing concepts defined by ISTQB [18].

Though the ISTQB foundation-level syllabus provides a solid foundation for understanding testing concepts and principles, it has certain obstacles when it comes to teaching practical testing skills. This paper aims to explore them and discuss the challenges of teaching testing effectively, with a focus on the ISTQB approach, and gives an outline to complete it. By understanding the challenges of using ISTQB materials in education we can identify ways to improve the teaching and learning of testing skills. Thus the paper tackles three research questions.

[1]https://www.istqb.org/

979-8-3503-3335-0/23/$31.00 ©2023 IEEE
DOI 10.1109/ICSTW58534.2023.00072

RQ1: To what extent is an ISTQB certification required or preferred in job applications within the software testing industry?
RQ2: How common is the teaching of software testing based on ISTQB principles within the curriculum of universities?
RQ3: What are the challenges of using ISTQB materials in education and how can they be overcome?

II. ISTQB Structure

The International Software Testing Qualifications Board (ISTQB) is a non-profit organization that provides learning materials and certifications for testers based on widely used testing principles and techniques in the industry. ISTQB has 67 member boards through which it is present in 130 countries of the world, and more than 836,000 individuals have at least one ISTQB certificate. Therefore, it is not surprising that many companies prefer testers who have ISTQB certificates. Of course, there are software industry sectors, domains, and also geographical regions in which this certification scheme is less important, but we can say with confidence that this is the de-facto industry standard in software testing.

ISTQB offers the Certified Tester Foundation Level (CTFL) certificate, which acts as a prerequisite for the other certifications. The AGILE, CORE, and SPECIALIST certifications cover a wide range of topics and skill levels, which is an excellent source for software testers to become an expert in specific fields of testing. Getting an ISTQB certification shows that a software tester has a strong understanding of the industry's best practices. It is also beneficial for testers who are looking to advance their careers or increase their earning potential.

Figure 1 shows the certificates a software tester can obtain and their relations (as of January 2023). For each certificate, ISTQB provides a syllabus and a couple of sample exams. After getting the Foundation Level certificate (CTFL), a software tester can take a path to be certified. All of these paths offer a chance for a variety of specializations.

An Agile certified tester has a deep knowledge of agile testing methodologies and the principles of agile software development. This individual understands the role of testers within an agile team and is able to support the team in planning and executing test-related activities. They adapt their testing experience and knowledge to align with Agile values and principles. Agile testers are highly valued by organizations that use Agile methodologies, as they help ensure the timely delivery of high-quality software products. Additionally, Agile-certified testers can provide valuable guidance and support to other team members in implementing Agile practices and achieving a successful Agile project outcome.

The Core certifications provide the skills needed to perform structured and thorough software testing across the software development life cycle. The Test Manager certification covers the knowledge and skills required to design an appropriate test approach for a project, build a test team, and develop the necessary testing competencies. Test Analysts learn how to translate software requirements and quality aspects into effective and efficient tests and gain an understanding of key testing techniques that help them do this translation. Additionally, Technical Test Analysts are trained in risk-based testing, white box testing, static and dynamic analysis, non-functional testing, and test automation; all things that are required to effectively implement and execute the different tests.

Specialist certifications provide in-depth knowledge and skills in specialized areas of software testing. These certifications cover the concepts, methods, and practices of specific areas such as game testing and mobile application testing. For instance, the Game Testing certification imparts the necessary understanding and skills to test games at all levels in game projects. The Mobile Application Testing certification equips testers with knowledge of the necessary mobile project activities, test processes, approaches, and automated test execution. These certifications are beneficial for testers who wish to expand their expertise in a specific field of software testing.

III. Job Applications

We looked for job applications on different job offering websites (Jobline [2], Profession [3], LinkedIn [4], EuroJobs [5]) to see how frequently ISTQB certification is required for the QA role. We provided the *"software tester", "QA", and "software engineer"* keywords in the search bar and manually examined the job applications. Note, that the Jobline and Profession portals are in Hungarian, so we needed to translate the keywords to get meaningful results. To be able to compare them, we set the location for international portals to Hungary, thereby ensuring that only Hungarian job postings were listed. Furthermore, LinkedIn is profile specific, i.e. it lists different job applications based on what information the user sets as experience and interests, thus it may vary from person to person. To avoid bias we used LinkedIn in incognito mode.

Figure 2 shows the number of jobs from each portal using the above-mentioned keywords. Since there is still a widespread preconception that testing is less important than designing software, it is not surprising that software engineers are sought after more than software testers. While analyzing job listings, we found that some software tester positions did not explicitly state a preference or requirement for an ISTQB certificate. However, these job listings did mention similar knowledge such as knowledge of black-and-white box testing techniques, and an understanding of software QA methodologies, tools, and processes. Although we did not classify these as positions that specifically favor having an ISTQB certification, it further supports our claim that the ISTQB Foundation Level certificate is commonly sought-after at entry-level software tester jobs. Additionally, it is important to note that obtaining an ISTQB certification not only demonstrates a certain level of knowledge and understanding

[2] https://jobline.hu/
[3] https://www.profession.hu/
[4] https://www.linkedin.com/
[5] https://eurojobs.com/

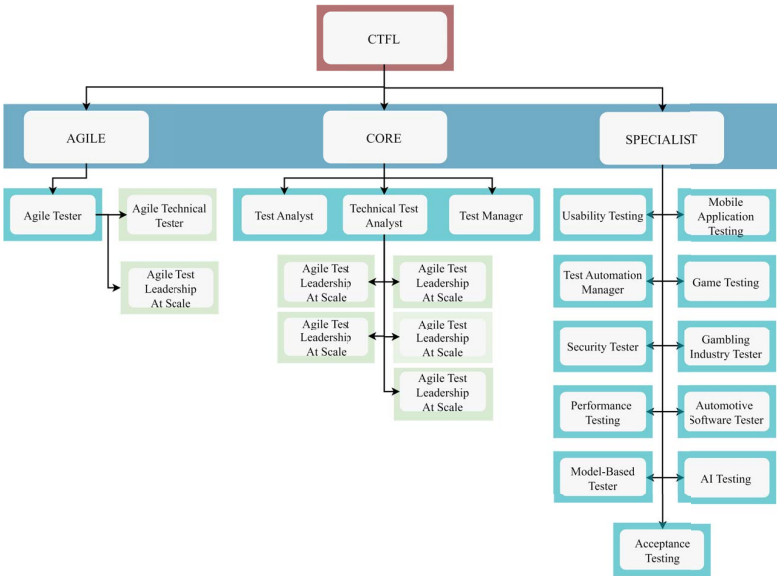

Fig. 1. ISTQB certifications

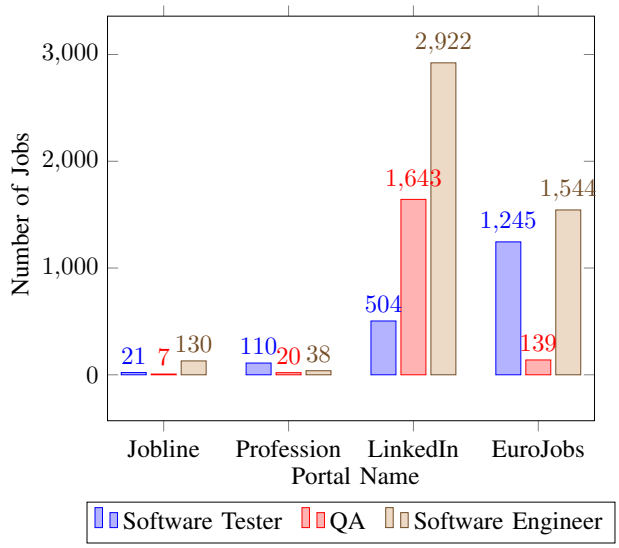

Fig. 2. Number of jobs in Hungary in 2023.
Keywords used: *Software Tester*, *QA*, *Software Engineer*

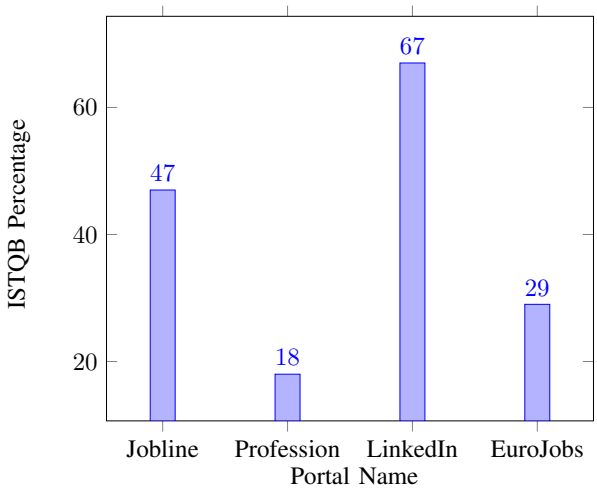

Fig. 3. Percentage of Software Tester jobs (out of a hundred) mentioning ISTQB as a requirement

of software testing but also shows an individual's commitment to professional development and career growth in the field.

Figure 3 shows the percentage of how frequently ISTQB is a requirement for Software Testing jobs. We found that all portals had more than 100 software tester job listings except for Jobline, which only had 21 job listings as seen in Figure 2. As a result, we manually investigated a maximum of one hundred job offerings from each site. When looking at the percentage of software tester jobs that require or prefer ISTQB certificates, we found that LinkedIn and Jobline had relatively high rates. Both of them are popular job listing websites, therefore, we can conclude that ISTQB certificates are commonly preferred by companies. While the other portals had a relatively lower rate of job postings where they favor the ISTQB certificate, it should not be interpreted as them favoring other certificates. Simply, there is no other software tester certification that could be equally sufficient and globally accepted. Furthermore, having an ISTQB certification can also open up opportunities for advancement and higher-paying positions within a company. It is also an international certification that is recognized globally. This supports our claim that ISTQB certificates provide the defacto knowledge that is required for software tester jobs.

Answer to RQ1: The results show that an ISTQB certification is generally favored in software testing job offers. Thus, we can conclude that the industry highly values the ISTQB certification.

IV. Test Education in University Curricula

The purpose of this section is to give a short survey of how software testing is taught in higher education. This is by no means a comprehensive survey, just an excerpt to illustrate the level to which testing is included in the curricula and especially how much ISTQB is mentioned in these. In addition to a literature review, we also checked several QS-ranked universities[6] for their curricula and software testing. It is worth mentioning that many universities do not provide the curricula publicly, only the requirements and topics they cover in the courses.

MIT offers open courses, and we found one single course that covers testing[7], but this course offers more content. It also covers basic Java skills, debugging, synchronization, version control, etc. The dedicated testing lesson covers white-box, black-box, unit testing, and code coverage, but it is not based on the ISTQB program. We found a few courses at Stanford that include testing, which are CS106B, and CS107. CS106B has a lecture[8] where they cover some aspects of testing. Similarly, CS107 provides a page[9] dedicated to software testing where they talk about black-and-white box techniques. We also found some courses at Carnegie Mellon University (CMU) that include teaching software testing. An example of that is the *15-112*[10] course that includes testing and debugging. Another example is *15-414*[11], an undergraduate course that is called *Bug Catching: Automated Program Verification.* Additionally, the course we found that is related to testing is *17-355*[12]. This course gives an in-depth knowledge of program analysis.

These courses generally teach some form of software testing as part of one or two lectures, however, they do not follow the ISTQB foundation-level material. Nevertheless, there is definitely evidence of the use of ISTQB syllabi at universities. For instance, the German Testing Board lists a number of such institutions on its website. [13].

Many universities consider including testing in their curriculum in Computer Science programs. Heckman et al. [15] integrated software testing principles in introductory core courses at North Carolina State University (NCSU). At the beginning of the semester, students were encouraged to test the assigned programs with various inputs. Later in the semester, white and black box techniques were introduced. Students were required to submit tests for their assignments. In the Software Development Fundamentals course, students received a more in-depth look at testing, learning about test levels, coverage, and static analysis. They were also required to perform testing for their development assignments. Other universities have similar modifications to their courses. [16], [17]

While incorporating testing in introductory programming courses is an excellent idea, it might be overwhelming for new students to learn that software development is more than just coding. Furthermore, they may not fully grasp the importance of software quality and find it difficult to accept that teaching software testing is as important as teaching software development. Nonetheless, it is essential for students to understand testing principles and how to avoid possible bugs. While teaching testing principles to computer science students throughout their education is important, universities should offer specialized software testing courses where students can learn testing concepts and principles in-depth.

The following articles tackle some approaches to how they effectively teach testing at their universities. Aniche et al. [19] made a pragmatic software testing education program at Delft University. Their course is currently a compulsory part of the Computer Science bachelor program. Their course covers topics from ISTQB industry certification. Lambers [20] used a sandwich approach for test education at the University of Potsdam. They first briefly address static analysis, and then they teach testing concepts and test automation, then they teach how to verify software. Krutz et al. [21] at the Rochester Institute of Technology offer a software testing course to Software Engineering, Computer Science, Computer Engineering, Electrical Engineering, and Game Design students. Buckley et al. [22] made a study with two undergraduate students from Florida Gulf Coast University. Their course is project-based and they used three different strategies over 4 semesters to teach black-and-white box techniques. Within these "strategies" students either had to test projects that were sponsored by local companies, a single project that was being developed during the semester, or a project that has been developed previously.

Balla et al. [18] made a study on the place of software testing in Hungarian higher education. They presented the state of software testing education in 4 Hungarian universities, and 3 of them actually build their software testing course on the ISTQB CTFL material. It is worth noting, that the curricula of these universities might have changed over the last 12 years since the publication of the article. One of the investigated institutions was the University of Szeged, whose software testing course we present in more detail in Section V. Budapest University of Technology and Economics (BME) offers a software testing course for Master's students. In the course, students are learning about the basic concepts and techniques of software testing, effective testing requirements, and possibilities of organizing an effective testing process. They are applying the basic software testing concepts described by the ISTQB. At Eötvös Loránd University (ELTE), software testing does not appear as a separate subject in

[6] https://www.topuniversities.com/university-rankings/world-university-rankings/2023

[7] https://ocw.mit.edu/ans7870/6/6.005/s16/classes/03-testing/

[8] https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1232/lectures/03-strings/

[9] https://web.stanford.edu/class/archive/cs/cs107/cs107.1234/testing.html

[10] https://www.cs.cmu.edu/~112/index.html

[11] https://www.cs.cmu.edu/~15414/s22/index.html

[12] http://www.cs.cmu.edu/~aldrich/courses/17-355-18sp/

[13] https://www.german-testing-board.info/en/universities/certified-tester-at-universities/list-of-universities/

their undergraduate curriculum. However, several subjects are concerned with testing methodology, the semantics of unit tests in different languages, and test environments. Two subjects in the Master's program deal with testing, one of which builds on the foundation-level ISTQB material and extends it with agile testing methodologies.

In addition, there is a dedicated software testing course[14] at the University of Debrecen. The topics of the course are publicly available, however, their lectures are not. The course covers a variety of techniques from the ISTQB CTFL syllabus and expands upon it by covering additional areas like testing databases and security testing, etc.

V. Test Education at University of Szeged

The Software Engineering Department at the University of Szeged, Hungary, offers a variety of courses that cover testing concepts for undergraduate students. Although most of the generic software engineering courses are rather focused on teaching software development and programming, they also include some form of testing, e.g. unit testing. Two specific software testing courses are offered to undergraduate Computer Science students, and one software testing course is offered to Master's students. All of these courses are elective, so students can choose to apply for them but they are not required to take it to graduate.

Fundamentals of Software Testing: The first undergraduate course gives lectures on the theoretical part of testing, alongside laboratory exercises where students gain practical knowledge by doing hands-on exercises. The theoretical lectures cover all content of the Foundation Level Syllabus of the ISTQB and give examples where needed, e.g. how to write test cases, how and where to report bugs, and how to write test plans. At the start of the semester, students in laboratories learn about testing concepts, e.g. how to write test cases, how and where to report bugs, how to write test plans, and need to select from a list of Open-Source programs from GitHub that they will thoroughly test during the term. The list consists of Java and C/C++ programs, which are ranked in the top 10 most popular languages in 2022 [15], therefore, they learn to use a variety of relevant testing tools.

By the middle of the semester, students learn about static analysis and white box testing. Students have to perform static analysis accompanied by code reviews or code coverage analysis as part of their assignment. They are taught to use Spotbugs, PMD, Checkstyle, and EclEmma for Java, and gcov for C++. Students may use other tools for their assignments.

By the end of the semester, students will have gained knowledge of black-box testing techniques. This allows them to effectively implement these techniques when creating test cases for their assignments. At the end of the semester, students are required to write a report that summarizes the methods and tools employed during testing.

[14]https://gyires.inf.unideb.hu/KMITT/c12/

[15]https://octoverse.github.com/2022/top-programming-languages#:~:text=Top%20languages%20used%20in%202022,place%20year%2Dover%2Dyear.

Practical Software Testing: The second undergraduate software testing course provides a practical introduction to software testing after the theoretical part, which students learn from the previous course. Additionally, the aim of the course is to learn about automation and to get a basic introduction to test automation tools and some agile testing practices. They learn how to use popular test automation tools/frameworks such as Postman, Selenium, JUnit, Mockito, Cucumber, etc., and get familiar with test-first approaches, like Test Driven Development and Behavior Driven Development. The lectures are held by guest lecturers invited from external companies who can use the tools at an expert level. Students are required to do weekly assignments using the tools presented in class. These tools are widely used in the industry, meaning that students who are mastering them will be in high demand when they enter the job market. By completing weekly assignments using the tools, students will gain hands-on experience, and if they have any problems they can ask for help from professionals.

Software Testing Methods: The Master's course covers the most important black-and-white box testing and static analysis methods, mainly from a technical point of view, and occasionally the scientific aspects are covered as well. This course aims to provide not just a basic discussion but an in-depth study of the subject, sometimes covering specific methods used in unusual circumstances or not yet widely used in the industry. These include mutation testing, combinatorial testing, test selection/prioritization, slicing, fault localization, etc.

Answer to RQ2: While testing is typically included in the courses for computer science programs in higher education, specialized testing courses that are based on ISTQB materials are not common.

VI. Challenges of Teaching Software Testing Based on ISTQB Syllabi

ISTQB certifications are globally accepted and are often requirements for companies, therefore, it is beneficial for students to learn the concepts and techniques the syllabi offer. Based on the literature review it can be seen that teaching software testing built around ISTQB syllabi is not widespread. However, based on our own experience as lecturers and on feedback from students, the approach to include the topics of this certification scheme is successful. Our recommendation to other higher educators is to consider including the topics covered by the ISTQB syllabi in their software testing courses.

However, this approach comes with some challenges as well, and in this section, we overview how to address them. ISTQB syllabi are designed to standardize testing principles, therefore, it is expected to be broad and let the tester figure out a few things for themselves, or complement the training on special practical projects. Many companies follow this path: they require the foundation level certification first from the employees as basic theoretical knowledge, and then they complement it with knowledge obtained from practical projects.

The syllabi intentionally do not mention what tools need to be used while testing. However, this is can be a challenge, since teachers need to be up to date with the usage of the most popular testing tools. It is also worth noting that the higher the certification level, the more theoretical the ISTQB syllabus becomes. For example, the Advanced Level Test Manager (CTAL-TM) material is built on test management, which is a very complex topic to teach in a classroom setting as it requires a testing team. This can make it difficult for educators to effectively cover all the material in a classroom setting. ISTQB's specialist certifications cover a number of current topics in the testing industry, but they may not fully address some unique challenges of testing, such as DevOps. As a result, other materials may be necessary.

We outline several challenges of using ISTQB syllabi as a base for testing education:

1) Highly theoretical: ISTQB syllabi mainly focus on teaching a theoretical understanding of testing concepts and principles. It is essential for testers to grasp these concepts, but they are not enough to use the knowledge in practice immediately.
2) Lack of hands-on exercises: It is fundamentally not designed to equip the student with hands-on exercises and testing tools. This can make it harder for students to generalize the concepts and use them in real-life environments.
3) Too broad: While it covers a wide range of testing concepts, it lacks in-depth information about using specific testing tools and techniques. This can make it difficult for students to gain experience in testing.

We demonstrate how we successfully addressed these challenges in our undergraduate software testing course mentioned in Section V, which builds around the basic knowledge required by the ISTQB CTFL syllabi.

A. To Overcome The Overly Theoretical Nature

First, we solved the highly theoretical problem by creating a laboratory component of the course, where students can learn how to create efficient test cases, bug reports, and test plans. This allows them to apply the theoretical concepts they have learned in a practical setting. They are introduced to these concepts using small, easy-to-understand real-life examples, and then gradually, they are required to perform quality assurance on real-life open-source programs. Additionally, the laboratory component provides students with hands-on experience that they can apply to their future careers in the tech industry. At the beginning of the semester, students learn how to create test cases on small examples like turning their screens on and off by following the well-known AAA pattern. The AAA pattern, also known as the Arrangement, Action, and Assertion pattern, is a widely used and natural way to structure test cases. [23] An example test case would look like Table I.

TABLE I
A TYPICAL TEST CASE FOLLOWING THE AAA PATTERN

	Test steps
Arrange	The screen must be turned on
Act	Manually push the power button on the monitor
Assert	The monitor's screen should be turned off

Apart from this small example, there are many more complex test cases in real life. However, students who had hardly any knowledge of testing before the course can learn a lot better from these small examples as they are easy to understand. These small examples serve as a foundation for the students to build upon as they learn to tackle more complex test cases in the course. Once they understand and can use the basic concepts, students are required to create test plans as their assignments on the chosen Open-Source program.

The test plan is an important document that guides all software testing projects. It outlines the tasks that need to be completed, the quality standards that need to be met, the resources required, the timeline, and the plan for managing risks. Students are advised to follow the IEEE 829 [24] Test Documentation Standard while writing their test plans. Though ISTQB has updated its reference from IEEE 829 to ISO 29119-3, we suggest using the prior for students, since it is shorter and easier to understand. We recommend that test educators adopt this approach as it simplifies the completion process for students and helps with grading the assignments for teachers.

B. Giving Hands-on Exercises

At the University of Szeged, we are teaching black box techniques in the later part of the semester. There are 5 black box techniques that the ISTQB CTFL [25] syllabus covers. While students may understand the importance of these techniques, they may have difficulty applying them in real-world situations. This may be due to a number of reasons, but one contributing factor could be that the techniques are based on specifications that are written in natural language. These specifications may be more complex and nuanced in real-world scenarios, making it challenging for students to effectively use the techniques they have learned. Therefore, we provide an anonymized specification of commercially used software for students to practice. One effective strategy to study black box techniques is to review the sample exams provided by ISTQB and practice solving questions that involve these techniques. Additionally, it is helpful for students to check if their answers are correct, as this allows them to identify any areas where they may need further improvement.

Along with this, we provide a set of examples and answers to practice these techniques, which can be found in the online Appendix [16]. Another recommendation we have is that students should be required to apply black box techniques while completing their testing assignments. This way, they can practice using these concepts on real-world specifications and become more proficient in their application. A lot of open-source projects have poorly written or no specifications, which is a challenge in itself. In that case, students may do exploratory testing with the help of the teacher before applying black box techniques.

[16]https://doi.org/10.6084/m9.figshare.21983024.v1

TABLE II
LIST OF PROGRAMS WE USE DURING OUR SOFTWARE TESTING COURSE

Tool Name	Purpose
GitLab	- Bug reporting - Write test plan - Store test case documentation
JUnit 5	- Unit testing - Integration testing - System testing
EclEmma	- Measure coverage for Java programs - Analyze the generated report
Gcov	- Measure coverage for C programs - Analyze the generated report
SpotBugs	- Static analysis - Find possible errors
PMD	- Static analysis
Checkstyle	- Static analysis - Catch coding styling violations

C. Using Specific Testing Tools

We overcome the third challenge by teaching the usage of testing tools and specific testing techniques. We show what we use in our course and provide an explanation for why we use them. Table II lists the programs we use in our course and their benefits. Students are required to use GitLab[17] for turning in their assignments which include bug reports, test plans, test cases, and so on. We used to use TestLink[18], an open-source test management system, for documenting test cases and plans, however, we decided to switch to GitLab since it makes it easier for the students that everything is in one place. This tool can be used to practice the test management part of the CTFL syllabus. We use JUnit[19] to teach several testing levels which complement the Test Levels part of the syllabus. This is beneficial for students since they are introduced to and taught how to properly use a tool that is commonly used in the industry.

For static software analysis, we used three different programs (SpotBugs[20], PMD[21], and Checkstyle[22]). All of these tools are commonly used in the industry and each has its own benefits. SpotBugs uses static analysis to look for bugs in Java programs. The benefit of using it is that it can help software engineers, including testers, identify potential bugs in their code before they are deployed, resulting in more stable and reliable software. Additionally, it can also help improve code quality by identifying patterns of poor coding practices. Similarly, PMD (Programming Mistake Detector) can identify patterns of poor code practices. PMD can also be easily integrated into build tools such as Maven and Gradle, allowing for automated code analysis as part of the development process. Additionally, the usage of Checkstyle is taught to the students. Checkstyle is a rule-based Java code checker, which automates verifying the code against coding standards. It is highly configurable and can be integrated into the git flow by using a git hook. This way the user is not able to commit anything until they resolve the code styling errors their code has. The students are required to review the code of their assignment programs, but they can use these tools as a guide. These tools complement the static analysis part of the ISTQB CTFL syllabus.

Lastly, we show how we use two separate coverage tools in education. EclEmma[23] is a code coverage tool for Eclipse. The benefit of using it is that it allows developers and software testers to easily see which lines of code are being executed during testing, and identify areas of the code that are not being sufficiently tested, enabling them to write more effective test cases. Similarly, Gcov[24] is a code coverage program for C/C++. It is used alongside with the GNU Compiler Collection (GCC) to analyze a program's source code and generate a report of which parts of the code have been executed and which have not. Both tools can generate reports on code coverage, which students have to analyze and draw conclusions from.

By explaining the solution to the three challenges the ISTQB syllabus has as educational material, we can answer the third research question.

Answer to RQ3: The ISTQB CTFL syllabus has some parts that induce challenges in the higher education context. We identified three during our university course: **highly theoretical, lack of hands-on exercises, and too broad**. We addressed them by using practical scenarios in exercises, including a set of tools used in specific environments.

VII. CONCLUSION

In this study, we examined how important ISTQB certifications are. First, in order to determine how crucial the certification is for software tester positions, we manually reviewed 321 job postings from 4 distinct job portals. We discovered that ISTQB certifications for software testers are mentioned in 18%–67% of the job offers in the different sites, and in 39% of all the job offers. Therefore, we came to the conclusion that ISTQB certificates are important and preferred in the industry.

Additionally, we looked at the curricula of different universities to see how software testing is integrated into education. Since many do not make their courses available to the public, we were limited to using the topics they teach. On the other hand, some universities published papers on how they are integrating software testing into their educational programs, however, they are not explicitly following the ISTQB syllabus. At the University of Szeged, we offer a specialized software testing course to Computer Science students. With our course, we follow the outline of the ISTQB CTFL syllabus. We identified some challenges when using it for education and we showed how we addressed them.

[17]https://gitlab.com/
[18]https://testlink.org/
[19]https://junit.org/junit5/
[20]https://spotbugs.github.io/
[21]https://pmd.github.io/
[22]https://checkstyle.org/
[23]https://www.eclemma.org/
[24]https://gcc.gnu.org/onlinedocs/gcc/Gcov.html

In conclusion, we can say that the ISTQB syllabi are good and useful, but one must be aware that it needs supplementary tasks when used as educational material. This is not surprising since the purpose of the syllabus is to help candidates pass the ISTQB exam by explaining techniques and not to prepare the student to use their knowledge in practice immediately. Despite the identified challenges, we want to advise universities to teach software testing based on ISTQB because students would learn relevant and useful techniques. Furthermore, if they decide to take the exam, they will already be familiar with the terminology and will have some prior knowledge, thus will not have to start from the beginning. A university educator who wishes to incorporate ISTQB into their software testing course should consider the following steps:

- Use the ISTQB syllabi as a guide for creating course content that covers relevant and useful techniques for software testing.
- Provide students with opportunities to practice applying the techniques they learn in real-world scenarios, such as through simulations (testing OS programs) or hands-on projects (See in online Appendix [25]).
- Seek ways to extend the basic syllabi with topics that are most relevant for the students in the education program in question.
- Continuously evaluate and adapt the course content to ensure that it aligns with the latest version of the ISTQB syllabi and meets the needs of the students.

In this paper, we presented what challenges we need to overcome using ISTQB CTFL syllabus for teaching software testing. However, further examination of advanced ISTQB syllabi remains an area of opportunity. Additionally, it would be interesting to compare with other subjects that follow a globally recognized educational material, like IREB [26] for requirements engineering, and teach them useful knowledge that can serve as a foundation if they want to attain certification.

ACKNOWLEDGEMENT

This research was carried out in project TKP2021-NVA-09 supported by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

REFERENCES

[1] J. C. Carver and N. A. Kraft, "Evaluating the testing ability of senior-level computer science students," in *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 2011, pp. 169–178.

[2] T. Shepard, M. Lamb, and D. Kelly, "More testing should be taught," *Communications of the ACM*, vol. 44, no. 6, pp. 103–108, 2001.

[3] J. Spacco, D. Hovemeyer, W. Pugh, F. Emad, J. K. Hollingsworth, and N. Padua-Perez, "Experiences with marmoset: designing and using an advanced submission and testing system for programming courses," *ACM Sigcse Bulletin*, vol. 38, no. 3, pp. 13–17, 2006.

[25] https://doi.org/10.6084/m9.figshare.21983024.v1

[4] V. Garousi, G. Giray, E. Tüzün, C. Catal, and M. Felderer, "Aligning software engineering education with industrial needs: A meta-analysis," *Journal of Systems and Software*, vol. 156, pp. 65–83, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121219301347

[5] O. A. L. Lemos, F. C. Ferrari, F. F. Silveira, and A. Garcia, "Experience report: Can software testing education lead to more reliable code?" in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2015, pp. 359–369.

[6] "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.

[7] "Iso/iec/ieee international standard - software and systems engineering - software testing – part 2: Test processes," *ISO/IEC/IEEE 29119-2:2021(E)*, pp. 1–64, 2021.

[8] "Ieee/iso/iec international standard for software and systems engineering–software testing–part 3:test documentation," *ISO/IEC/IEEE 29119-3:2021(E)*, pp. 1–98, 2021.

[9] "Ieee/iso/iec international standard - software and systems engineering–software testing–part 4: Test techniques," *ISO/IEC/IEEE 29119-4:2021(E)*, pp. 1–148, 2021.

[10] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons: New York, NY, 2011.

[11] C. Kaner, J. Bach, and B. Pettichord, *Lessons learned in software testing*. John Wiley & Sons: New York, NY, 2002.

[12] G. M. Weinberg, *Perfect Software and other illusions about testing*. Dorset House: New York, NY, 2008.

[13] P. C. Jorgensen, *Software Testing: A Craftsman's Approach*. Auerbach Publications, 2008.

[14] L. Copeland, *A Practitioner's Guide to Software Test Design*. USA: Artech House, Inc., 2003.

[15] S. Heckman, J. Y. Schmidt, and J. King, "Integrating testing throughout the cs curriculum," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2020, pp. 441–444.

[16] M. H. Goldwasser, "A gimmick to integrate software testing throughout the curriculum," *ACM SIGCSE Bulletin*, vol. 34, no. 1, pp. 271–275, 2002.

[17] J. L. Gersting, "A software engineering "frosting" on a traditional cs-1 course," in *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education*, 1994, pp. 233–237.

[18] K. Balla, Á. Beszédes, B. G. Csonka, T. Heckenast, and A. Kovács, "The software testing curriculum in the hungarian education in conjunction with international standards," in *Proceedings of the Conference on Informatics in Higher Education 2011 (IF 2011)*. Debreceni Egyetem, Informatikai Kar, Aug. 2011, pp. 1096–1103.

[19] M. Aniche, F. Hermans, and A. Van Deursen, "Pragmatic software testing education," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 414–420.

[20] L. Lambers, "How to teach software testing? experiences with a sandwich approach," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. IEEE, 2020, pp. 425–428.

[21] D. E. Krutz, S. A. Malachowsky, and T. Reichlmayr, "Using a real world project in a software testing course," in *Proceedings of the 45th ACM technical symposium on Computer science education*, 2014, pp. 49–54.

[22] I. Buckley and P. J. Clarke, "Experiences of teaching software testing in an undergraduate class using different approaches for the group projects," in *2021 ASEE Virtual Annual Conference Content Access*, 2021.

[23] C. Wei, L. Xiao, T. Yu, X. Chen, X. Wang, S. Wong, and A. Clune, "Automatically tagging the "aaa" pattern in unit test cases using machine learning models," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–3.

[24] "Ieee standard for software and system test documentation," *IEEE Std 829-2008*, pp. 1–150, 2008.

[25] I. S. T. Q. B. (ISTQB), "Cerfified tester foundation level syllabus – version 2018 v3.1," 2018.

[26] K. Pohl, *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam – foundation level – IREB compliant*. Rocky Nook, Inc., 2016.