Teaching Software Testing to Industrial Practitioners using Distance and Web-Based Learning

Eduard Paul Enoiu

Mälardalen University, Västerås, Sweden
eduard.paul.enoiu@mdh.se

Abstract. Software testing is a business-critical process used by private and public organizations and an important source of market competitiveness. Employees of these organizations are facing tough competition and are required to be able to maintain and develop their skills and knowledge in software testing. In the education market, many commercial courses and certifications are available for industrial engineers who wish to improve their skills in software development. Nevertheless, there is a lack of access to world-leading research within the software testing field in these commercial courses that supports the companies' innovation in software testing. As an alternative, universities are approaching this challenge by developing academic courses on software testing that can suit professionals who need to be able to combine work and studies. This study highlights several good approaches and challenges in developing and teaching three distance web-based software testing courses targeting test practitioners. The proposed approaches for enhancing teaching of software testing in an online setting for industrial practitioners are: active participation at the student's pace, inclusion of software testing artifacts from the student's organization as part of assignments, continuous access to online materials, the use of short video materials on testing theory, and setting clear expectations for performing online test design assignments. Finally, several challenges have been identified: poor feedback on assignments, distances between students and teachers, the use of non-realistic assignments and the difficulty for industrial practitioners to complete academic assignments each week. Future work is needed to explore these results in practice, for example on how to shorten distances between students and teachers, as well as how to enhance the inclusion of real-world testing artifacts in course assignments.

Keywords: software testing education · web-based learning · online education · industrial practitioners · software engineering education

1 Introduction

Software plays a vital role in our daily lives and can be found in a number of domains, ranging from mobile applications to medical systems. The emergence and wide spread usage of large complex software products has profoundly influenced the traditional way of testing software. Nowadays, organizations need skilled engineers that should maintain their software testing knowledge throughout their careers. Even if many online courses on software engineering[1] and testing[2] are available from both private and public sectors, there is a lack of evidence on how to design and teach such courses, what particular challenges teachers are facing and how these courses can be improved and tailored to industrial practitioners [11]. Specifically, software testing education [2] is an important aspect of a thorough software engineering education. In these courses, students learn to apply specific test design techniques and technologies for a given software under test.

In this paper we present the results of a longitudinal study on three online software testing academic courses targeting industrial practitioners[3]. We identified several challenges that should be taken into account when designing online courses on software testing: poor feedback on assignments, distances between students and teachers, poorly supported assignments and some industrial practitioner specific challenges. In addition, we identified several good approaches for enhancing the development of online courses in software testing: active participation at the student's convenience and pace, inclusion of software testing artifacts from industrial practice and creation of short video lectures of software testing theory. These results can be used to improve online courses in software testing that can suit industrial practitioners who need to be able to combine work and studies.

2 Software Testing Education

Software testing education is an important aspect of learning software engineering. Education in software testing should ensure the supply of software testing knowledge. Testing is widely considered to be an under-prioritized activity in software and systems development. Courses in testing are providing an understanding of the fundamental problems, as well as practical methods and tools for a systematic state-of-the-art approach to software testing. Books such as the one written by Ammann and Offutt [1] are used in teaching as a pedagogical approach to software testing instruction method in many university courses.

Garousi et al. [8] performed a literature review to map the the topic of software-testing education. There are many pedagogical approaches and specific tools used in testing education. For example, Hynninen et al. [13] performed a survey to find out the current state of industry in software testing education. According to their results, some key learning objectives in software testing disciplines can be used to identify the knowledge expectations from university

[1] For example Edx and Coursera offer MOOC-based software engineering courses.
[2] Udemy is one of the providers that offer software testing MOOC courses.
[3] The courses are given within the frameworks of PROMPT (`https://www.promptedu.se/`) and FUTURE (`https://www.mdh.se/en/malardalen-university/research/research-projects/futuree`), cooperation projects between academia and industry with the aim of strengthening competitiveness in Swedish companies.

graduates and to align with the industry requirements. Garousi et al. [8] has also identified the ways how to overcome some of these challenges in testing education, including the alignment with industry needs. Nevertheless, there is a lack of distance and online education course offerings for industrial practitioners. In this paper we aim to cover this subject and identify the challenges and good approaches in teaching software testing to industrial practitioners.

3 Moving to Distance and Web-Based Learning for Industrial Practitioners

In the last five years, we have started digitizing campus courses and developing new online courses to be exclusively given to industrial professionals. Working in this new setting involves a close interaction with students using digital tools. Nowadays, teaching has evolved and teachers are adopting blended learning techniques throughout the whole process of giving a course. Similarly to the results outlined in the study of Garrison et al. [9], we acknowledge that exploring and assessing the impact of blended learning is important in bringing more relevant learning experiences in these online courses. In the following sections, we outline three online courses on software testing given at Mälardalen University.

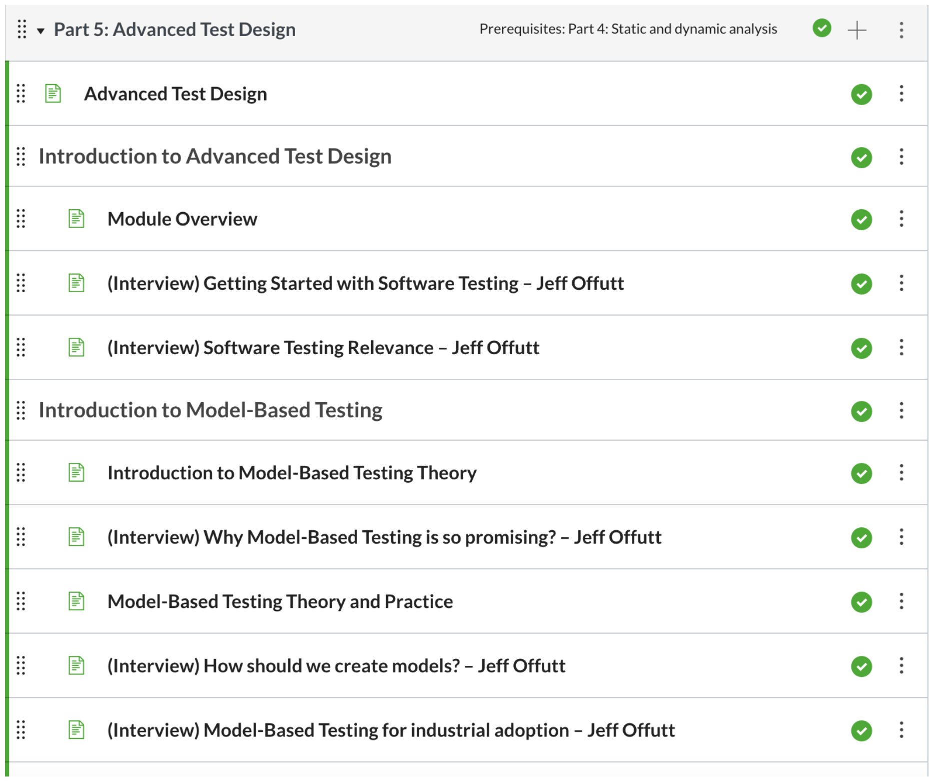
3.1 A Course Module on Advanced Topics in Software Testing

This course developed at Mälardalen University provides an understanding of the fundamental problems, as well as practical methods and tools for a systematic state-of-the-art approach to software testing [4]. After the course, the participants are expected to have an overview knowledge in more advanced testing methods (such as model-based testing, mutation testing and search-based testing), and in the state-of-the-art in software testing research. As shown in Figure 1, the course is given in a flexible format where the theoretical content, covered in video lectures, is interleaved with practical exercises. The course is divided into five 1.5-credit modules: Introduction to software testing and test design, unit testing, test design and automation, testing at integration and system level, static and dynamic analysis and advanced test design (the course module under investigation).

3.2 Model-Based Testing Course

This online course deals with model-based testing[5], a class of technologies used to assess the quality and correctness of large software systems in a more efficient and effective way than traditional testing methods. Throughout the course the participants learn how to design and use model-based testing tools, how to create realistic models and how to use these models to automate the testing

[4] https://www.promptedu.se/quality-assurance-the-applied-science-of-software-testing
[5] https://www.promptedu.se/quality-assurance-model-based-testing-in-practice/



Part 5: Advanced Test Design Prerequisites: Part 4: Static and dynamic analysis

Advanced Test Design

Introduction to Advanced Test Design

Module Overview

(Interview) Getting Started with Software Testing – Jeff Offutt

(Interview) Software Testing Relevance – Jeff Offutt

Introduction to Model-Based Testing

Introduction to Model-Based Testing Theory

(Interview) Why Model-Based Testing is so promising? – Jeff Offutt

Model-Based Testing Theory and Practice

(Interview) How should we create models? – Jeff Offutt

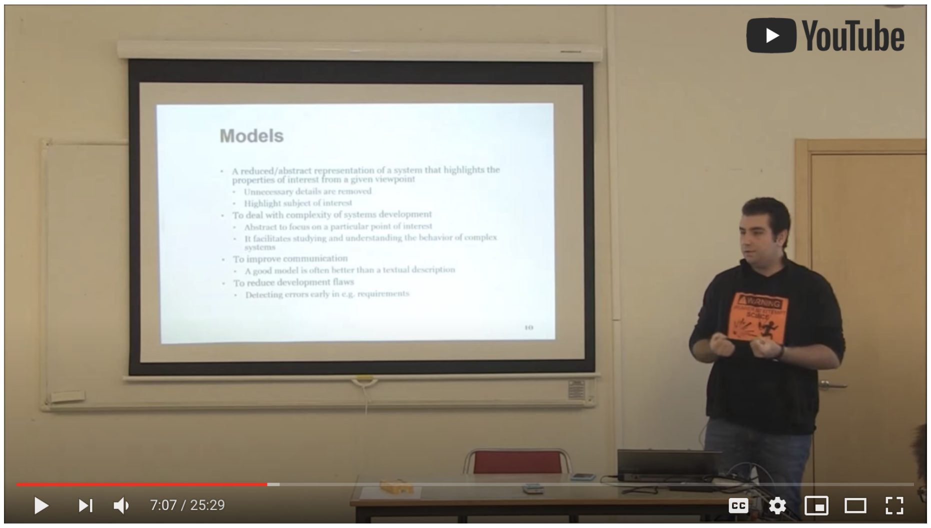
(Interview) Model-Based Testing for industrial adoption – Jeff Offutt

Fig. 1. An overview of the course material as part of the advanced test design module containing video lecturers and interviews with experts.

process in their organisation. We developed online material (including recorded videos of certain lectures as shown in Figure 2) on basic model-based testing terminology, creating models from industrial code, executing model-based tests and how to use model-based testing in practice. After completing the course, the students are expected to have acquired knowledge about models and understand model-based testing, develop practical skills and abilities on applying model-based testing in industrial practice, to test software using model-based testing in structured, organised ways. The students are admitted based on both specific entry requirements for credits in computer science and industrial experience. The students can apply for the course and get their eligibility evaluated based on knowledge acquired in other ways, such as work experience, and other studies.

3.3 Automated Test Generation Course

The automated test generation course is focusing on how to generate tests automatically in the sense that test creation satisfying a given test goal or given



Models

- A reduced/abstract representation of a system that highlights the properties of interest from a given viewpoint
  - Unnecessary details are removed
  - Highlight subject of interest
- To deal with complexity of systems development
  - Abstract to focus on a particular point of interest
  - It facilitates studying and understanding the behavior of complex systems
- To improve communication
  - A good model is often better than a textual description
- To reduce development flaws
  - Detecting errors early in e.g. requirements

10

Fig. 2. A video lecture on model-based testing recorded using a campus course set-up.

requirement is performed automatically [6]. This course provides an understanding of automating software testing using program analysis with the goal of intelligently and algorithmically creating tests. The course covers search-based test generation, combinatorial and random testing while highlighting the challenges associated with the use of automatic test generation. The student is learning about how to automatically generate test cases with assertions and to have a working knowledge and experience in static and dynamic generation of tests. The course is using a learning management system based on a discussion forum (as shown in Figure 3) in which students and teachers collaborate with each other throughout the course.

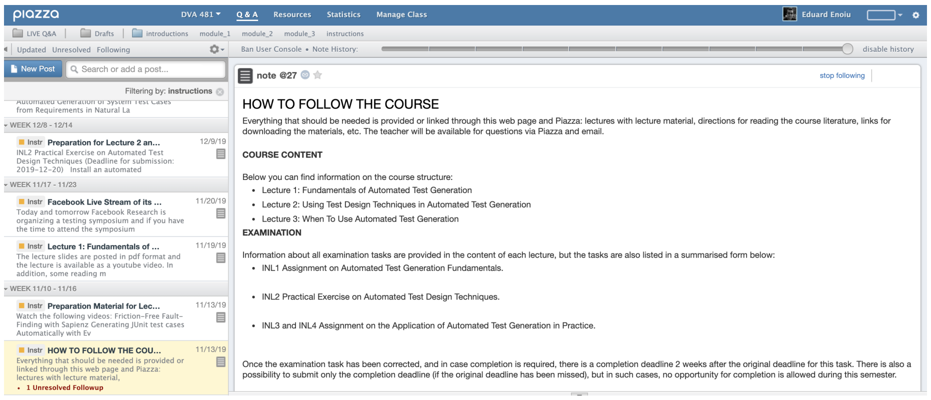
4 A Longitudinal Study of Developing and Teaching Three Online Courses in Software Testing

We present our experiences from developing and running three online courses (described in Section 3) during the last five years. The courses were offered as individual courses focused on flexible learning especially suited for working professionals with a large part of the teaching being web-based.

4.1 Case Study Methodology

Little longitudinal research has examined online teaching in software engineering or software testing. This study helps fill this gap. The approach uses a case

[6] `https://www.mdh.se/utbildning/fortbildning/ai-och-mjukvaruutveckling/futuree-automated-test-generation`



Automated Generation of System Test Cases from Requirements in Natural La

WEEK 12/8 - 12/14

Instr **Preparation for Lecture 2 an...** 12/9/19
INL2 Practical Exercise on Automated Test Design Techniques (Deadline for submission: 2019-12-20) Install an automated

WEEK 11/17 - 11/23

Instr **Facebook Live Stream of its ...** 11/20/19
Today and tomorrow Facebook Research is organizing a testing symposium and if you have the time to attend the symposium

Instr **Lecture 1: Fundamentals of ...** 11/19/19
The lecture slides are posted in pdf format and the lecture is available as a youtube video. In addition, some reading m

WEEK 11/10 - 11/16

Instr **Preparation Material for Lec...** 11/13/19
Watch the following videos: Friction-Free Fault-Finding with Sapienz Generating JUnit test cases Automatically with Ev

Instr **HOW TO FOLLOW THE COU...** 11/13/19
Everything that should be needed is provided or linked through this web page and Piazza: lectures with lecture material,
- 1 Unresolved Followup

HOW TO FOLLOW THE COURSE

Everything that should be needed is provided or linked through this web page and Piazza: lectures with lecture material, directions for reading the course literature, links for downloading the materials, etc. The teacher will be available for questions via Piazza and email.

COURSE CONTENT

Below you can find information on the course structure:
- Lecture 1: Fundamentals of Automated Test Generation
- Lecture 2: Using Test Design Techniques in Automated Test Generation
- Lecture 3: When To Use Automated Test Generation

EXAMINATION

Information about all examination tasks are provided in the content of each lecture, but the tasks are also listed in a summarised form below:
- INL1 Assignment on Automated Test Generation Fundamentals.
- INL2 Practical Exercise on Automated Test Design Techniques.
- INL3 and INL4 Assignment on the Application of Automated Test Generation in Practice.

Once the examination task has been corrected, and in case completion is required, there is a completion deadline 2 weeks after the original deadline for this task. There is also a possibility to submit only the completion deadline (if the original deadline has been missed), but in such cases, no opportunity for completion is allowed during this semester.

Fig. 3. A view of the forum learning platform used in the automated test generation course.

study design methodology informed by a number of methods from ethnography[7], including the use of repeated surveys, anchored in discussions of artefacts created by the course responsible. We followed the qualitative approach recommended by Calderhead [6] in the form of a case study for gaining insights into teaching through a variety of data collection methods over time. This study was conducted as a multiple case study comprising three individual case studies, each focusing on one of the courses described in Section 3. The research questions are: (1) What challenges are affecting online teaching of software testing for industrial practitioners? (2) What good approaches for improving distance and web based learning of software testing are found when teaching industrial practitioners?

This study takes a qualitative approach, focusing on the experiences of the course responsible and incorporating the knowledge of the context and its insights and perceptions, while attempting to better understand online teaching of software testing. The research was undertaken longitudinally to identify and interpret the commonalities in each course instance from spring semester 2015 until spring 2020. Data was collected through analysis of study materials at the end of each course by focusing on learning objectives, learning activities and assessment tasks. No personal data was collected during this process. Data analysis was conducted to generate hypotheses and broader concepts [3] that can be used by both researchers in investigating online teaching as well as by teachers that can include these results in their own courses.

Methodological Approach for Data Collection. As ethnography aims to explore a particular social phenomenon through research methods such as observations, we used this approach for data collection. This allowed the researcher to gather data within the course environment (the context of the study). We

[7] Ethnography is a research method for gathering data by active participation in the studied phenomenon [14].

captured and interpreted the experiences of the students. The researcher observed the activities in these course settings, taking handwritten notes. These activities included interactions between the teachers and the learners as well as the development of each course.

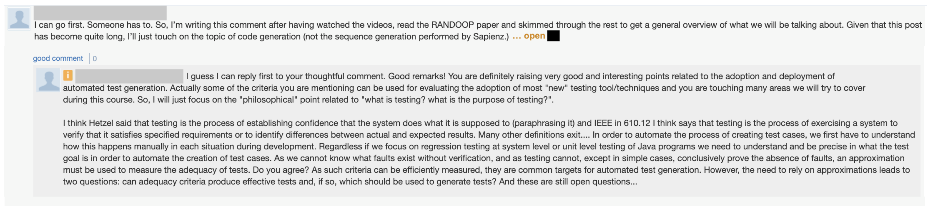
Data Analysis. We employed a process of thematic analysis to identify themes emerging from the data set (i.e, researcher's observations and personal reflections). This data was coded manually by grouping data (i.e., in segments of text) under individual codes. These codes were organized into two categories: good approaches and challenges. In addition, from the ensuing categories two other overarching themes emerged: teaching and learning software testing with digital tools and asynchronous communication between students.

4.2 On Teaching and Learning with Digital Tools

In practice, in the three courses under study, we are delivering online content outside of the virtual classroom and students can watch online lectures in advance, have online discussions at home while learning new concepts in the virtual classroom with teacher guidance through step-wise progression. This is achieved by tackling content complexities with digital assignments that support progressive learning [12]. During this longitudinal study we have learned how to use a variety of pedagogical techniques, including the use of visual aids (e.g., dynamic diagrams, interactive slides, videos and audio recordings), virtual group-work, student presentations and use case discussions. In addition, the use of learning platforms such as Piazza[8], Scalable Learning[9] and Canvas[10] is important in using the teaching aids in an efficient and effective manner. We have developed material for online courses, including:

– Course Marketing Material. (e.g., Automated Test Generation[11], Software Testing in the Video Game Industry[12], Software Testing Course[13])
– Audio Podcast. In 2017 we have started a podcast called Testing Habits[14] in which we have conversations with researchers, scientists and technologists about technology transfer, software testing and software engineering research. We have used this audio material in all ongoing courses.
– Video Recordings of Lectures. We have developed video recordings of regular lectures to be used in the flipped classroom method (e.g, Model-Based Software Testing Lecture[15]).

[8] `https://piazza.com/`
[9] `https://www.scalable-learning.com/`
[10] `https://www.instructure.com/canvas/higher-education/platform`
[11] `https://play.mdh.se/media/0_f1243mpq`
[12] `https://www.youtube.com/watch?v=CBIhu_9OolY`
[13] `https://www.youtube.com/watch?v=OR63w8Iod9I`
[14] `https://podcasts.apple.com/us/podcast/testing-habits/id1255653631`
[15] `https://www.youtube.com/watch?v=4fXBrZnj2JU`


[redacted]
I can go first. Someone has to. So, I'm writing this comment after having watched the videos, read the RANDOOP paper and skimmed through the rest to get a general overview of what we will be talking about. Given that this post has become quite long, I'll just touch on the topic of code generation (not the sequence generation performed by Sapienz.) ... open

[redacted] I guess I can reply first to your thoughtful comment. Good remarks! You are definitely raising very good and interesting points related to the adoption and deployment of automated test generation. Actually some of the criteria you are mentioning can be used for evaluating the adoption of most "new" testing tool/techniques and you are touching many areas we will try to cover during this course. So, I will just focus on the "philosophical" point related to "what is testing? what is the purpose of testing?".

I think Hetzel said that testing is the process of establishing confidence that the system does what it is supposed to (paraphrasing it) and IEEE in 610.12 I think says that testing is the process of exercising a system to verify that it satisfies specified requirements or to identify differences between actual and expected results. Many other definitions exit.... In order to automate the process of creating test cases, we first have to understand how this happens manually in each situation during development. Regardless if we focus on regression testing at system level or unit level testing of Java programs we need to understand and be precise in what the test goal is in order to automate the creation of test cases. As we cannot know what faults exist without verification, and as testing cannot, except in simple cases, conclusively prove the absence of faults, an approximation must be used to measure the adequacy of tests. Do you agree? As such criteria can be efficiently measured, they are common targets for automated test generation. However, the need to rely on approximations leads to two questions: can adequacy criteria produce effective tests and, if so, which should be used to generate tests? And these are still open questions...

Fig. 4. An example of a synopsis and evaluation conversation during a reading assignment.

– Interviews with Experts. We have developed video interviews with relevant academics to be used during lectures[16]. Voice over Slides Material. Some of the videos we developed are in the form of voice over slides (e.g., Lecture on Requirement based Testing[17]).

During the last five years we observed that continuous teaching in online courses is a source for the teacher's digital learning evolution. The learning evolves especially when students ask direct questions and through the correction of online assignments. This is part of continuously improving the video lectures and assignments by directly reflecting on how students are able to solve them in an innovative way.

4.3 Asynchronous Communication and Course Assignments

For all three online courses most of the class is taught asynchronously and assignments leverage online educational opportunities to virtually merge students. All groups in these case studies are using the same learning platform, watch the same lectures, and read and discuss testing subjects. One example of such a specific activity is an assignment in which students are summarizing, evaluating and critiquing a certain scientific paper on software testing. In this chosen assignment, one student examines and critiques certain software testing papers while others have the opportunity to critique the original synopsis. A snapshot of a part of such a synopsis is shown in Figure 4. For the set of papers, each student must submit a simple evaluation. The evaluation should be about a page long and must be posted on the learning platform. Several people will be assigned to write and post a synopsis evaluation for each paper. The synopses should: (1) describe the testing problem, (2) describe the goals of the paper, (3) describe and analyze the testing technique presented or the experimental setup, (4) discuss the results and (5) analyze the industrial applicability. Two students are designated as "skeptics" for each paper and are required to disagree with the posted synopsis and provide reasonable arguments. All students can join in the discussion for each paper throughout this assignment. All postings, including

[16] `https://www.youtube.com/watch?v=jbrhbnmkyAI`
[17] `https://www.youtube.com/watch?v=1pnqEom-xOw`

the reviews are done using the learning platform and all students can join the discussion and read these comments.

Since this assignment is supposed to provide a collaborative learning opportunity and the discussions are asynchronous, the aim is to focus on the learner and not the teacher, implying a more reliable understanding of the content [4]. In this way we aim to directly improve learning by focusing on the outcomes students are meant to deliver during the assignment. This focus opens up different perspectives on how knowledge and learning is transformative in these strict conditions of time and space. One perspective on learning achieved during several assignments during the last five years is related to the theory of collective activities to produce scientific knowledge [16]. Roth and Lee argue that knowledge and learning from scientific articles as a collective praxis process achieve more advanced forms of knowledge than individual could produce. This implies that an assignment that tries to create opportunities for literacy that emerges from a collective activity can be used to develop communities of praxis inside a course. Another highly-related perspective of knowledge and learning relates to Dewey's pragmatism [15]. This is where knowledge takes action form and theories (i.e., papers and reports read by students) become a tool and the application of such a tool (i.e., critique work during the assignment) is producing and reorganizing the experience needed for obtaining more understanding. In addition, we observed during multiple course instances that the chosen assignment relies on the problem-solving abilities of the students involved in this process.

These two perspectives of pragmatism and praxis on this kind of assignments can be used to focus on different aspects of quality in learning and teaching. Since online courses attract larger cohorts, the emerging community contained in some cases unengaged students and lack of interactivity between them. Assignments from this perspective can result in ineffective assessment and relatively limited feedback. To counteract this, some researchers [7] have proposed new engagement strategies for community learning by using more interactive digital content and gamification. Given the environment and online platform used, there are ways in which the pragmatic perspective can support learning and development for this particular assignment. The issue here is that our own experiences in using these assignments are not profound given that the assignment under study has been tested in practice in couple of instances of the course. There is a scant opportunity for improving it, but there is some evidence [18] that more analysis of these experiences together with constructs of pragmatic interactions can be used for a proper analysis. This is especially needed when taking into account that the technology used can influence the context of pragmatic learning.

Our experience is that when designing software testing assignments the teacher interaction and reflection is important when providing learning opportunities for feedback and critical review. We discovered that ceding control of learning and knowledge acquisition to the learner can be formative and encourage community reflection. In the end, our scope is to develop a small community in an online setting which is a challenging proposition in itself since learning is a social process made up of connections between the concepts learned and the other learners in the community. This assignment is constructively aligned with the rest of the course since the other assignments are built on stepwise analysis and reflection by the students. The peer review assignment was used as a method for this particular learning activity and the course is all about learning how to be a tester. The link between the other assignments and this one is strong given that only a successful engagement in the activities and completion of the assessment task can make a student achieve the subsequent individual learning outcomes related to peer review and experimentation.

In addition, the peer review activities used in all three courses can be redesigned and improved using the behaviorist theory of teaching [5]. The idea would be to use both positive and negative reinforcement to shape the online behavior of students by focusing on blended learning. The objective of the redesign should focus on creating a more dynamic community where active participation and teacher reinforcement can be implemented by using awards and gamification aspects through competitions. In these sessions, the community members can realize that they are not alone in experiencing the reinforcements and lead to a sense of trust in the group and their achievements.

4.4 Challenges and Good Approaches

In this section we outline the identified challenges affecting online teaching of software testing for industrial practitioners. In addition, we present several good approaches for improving distance and web based learning of software testing. We collected these results based on our experiences from teaching several instances of these courses in the last five years using an ethnography-based case study and thematic analysis.

The identified challenges are outlined in Table 1. For all three courses we experienced that poor feedback on assignments can influence the progression of the student throughout the course. One recommendation is to define a clear success criteria and how you as a teacher are going to measure it in an online setting. Getting the students to keep going throughout the course, means that the teacher needs to ask for feedback early and give feedback early and often. To address this challenge we used audio and video assignments feedback instead of a traditional source code and tests solution. In this way, the student could rather quickly gather evidence needed to modify the assignment for passing the stated learning outcome.

Another challenge relates to the geographical, social, cultural, and temporal distances that can have an impact on online teaching. These distances are not well studied in the teaching of software testing literature. We experienced concerns from students that factors related to these distances are making communication between teacher and student more difficult. For example, we observed that spoken language in videos affects the communication of different testing theory subjects. In our experience, a way of overcoming this type of distances is to use different tools and techniques such as filming a practical test session in the test environment while performing a certain assignment. However, the use of testing and programming tools can negatively impact learning efficiency.

Table 1. Overview of challenges and recommendations for teachers.

Challenge	**Recommendations**
1. Poor feedback on assignments	Speed up feedback loops by providing audio or video feedback.
2. Distances between students and teachers	Short distances improve online teaching of software testing. Some distances can be mitigated with certain tools.
3. Testing topics are not sufficiently explored in certain assignments	Use realistic testing assignments that includes uncontrolled variables which creates the complexity necessary to fully understand the practical testing challenges involved in a certain topic.
4. Poorly supported assignments by other teaching means	Assignments need to be clearly connected to the testing-related lecture material, but also to the reading instructions.
5. Difficulty for industrial practitioners to complete academic assignments each week.	Assignments need to be relevant for the company and the student.
6. For industrial practitioners completing the course is not as desirable as it is for regular students.	Learning effects are more important and students need to be engaged for visible results throughout the course.

The challenge is that these tools may have compatibility distances and may not always work well given that some are used in ways they are not designed for.

Just as in empirical and experimental software research, the use of controlled experiments as assignments where we are able to control all independent variables in order to measure the dependent variables seems needed. The challenge we have faced (Challenge 3 in Table 1) relates to cutting away all that which is not relevant in an assignment, and hence the solution was easier to comprehend. For some classes of testing assignments (e.g., test design techniques theory) we found that this is not a problem, whereas for other more hands-on assignments we found that developing realistic scenarios is a challenge. Our recommendation is to include assignments that contain uncontrolled variables in which the actual complexity is needed to fully understand the practical challenges involved in a testing topic. For example, software testing courses are often taught with a focus on test design techniques and test execution, whereas the real challenges arises from software testing ”at large” such as in the case of model-based testing and automated test generation, where intelligent and automated methods need to be used when thousands of new code changes happen per week and ambiguously specified test specifications are used. Clearly, the latter situation calls for different pedagogy methods and more realistic assignments to be created.

Another challenge relates to assignments that are loosely connected to the lectures and the course book. Some students were not able to directly identify the material that is required to understand what an assignment is about and how to conduct it. In an online setting, a practitioner has a limited time to perform an

Table 2. Overview of good approaches and their implications for teachers.

Approach	**Recommendations**
1. Active participation in teaching at the student's convenience and pace (although within a specified time frame).	Encourage collaboration given that learners are strongly motivated to acquire new testing skills and make professional progress. The learning activities need to be flexible in time, place and form to cater for different learning styles and circumstances among the involved testing practitioners and their companies.
2. Inclusion of software and testing artifacts from the student's organization as part of the course assignments.	Testing education for professionals with employment in software companies needs context-specific learning elements by choosing situated learning.
3. Continuous access to key concepts and online course materials.	Motivate your students by providing mandatory and optional materials. Make sure to link all the related material explicitly and throughout the course.
4. Video segments on testing theory should be restricted to a few minutes.	Human attention span is short and hence your recorded video segments should be restricted to a few minutes (at most 15-30 minutes).
5. Provide clear expectations together with the needed tools and frameworks needed for performing assignments.	Use suitable tools and make sure they can be used by all students given that you communicate clear expectations.

assignment. A recommendation for addressing this challenge is to design online assignments that are clearly connected to both the lecture material and also to the reading instructions.

In all three courses we observed that it is more difficult to motivate industry practitioners to continuously perform an academic assignment. As a recommendation, a teacher in software testing needs to design assignments that are relevant for the company and the student. In addition, the assignment needs to contain precise instructions throughout the entire assignment. Another identified challenge relates to industrial practitioners' lack of desire to obtain credits and completing all parts of a course. In our online courses, we observed that obtaining credits and completing the course were not important reasons for performing certain assignments. Some students did not complete a certain assignment on time and drop out of the course. As a recommendation, teachers might not want to maintain strict deadlines during the course.

In addition, we identified several good approaches for online teaching of software testing (shown also in Table 2). A first approach is to encourage active participation by following the actual pace of the student. A second good approach relates to the inclusion of software artifacts from the student's context in the course assignments. This is important, since different learning goals require different ways of teaching and learning. Testing examples and exercises should connect to the students' companies and application domains. A third good approach is focusing on providing, throughout the course, all the material including concept maps as inventories for all course modules. These are powerful tools for outlining and relating key testing concepts. In addition, we included stories since these tend to be remembered better than facts or abstract principles. Given that these courses were separated in modules, we spaced out the study material since, in our experience, this leads to better long-term learning than providing everything in a single module. A fourth good approach identified in several course instances is to make sure that students can actually follow lectures. One should record video segments on software testing theory that are not longer than a few minutes. It is more important to give an easily accessible overview of the testing topic that can continue with an in-depth exercise. In addition, a fifth good approach is to provide clear expectations on each assignment and use tools that can be used easily by all students. It is important that each module has a workload breakdown structure for all expectations per week, including how communication should take place (e.g., in discussion groups).

5 Discussions and Limitations

Overall, the results from this study are obtained using the experiences in teaching software testing in an online setting targeting industrial practitioners in three separate courses. Even if these results are directly related to online education in software testing, some of the listed challenges and good approaches are generic enough and can target any online course. We recommend evaluating these challenges and good approaches in teaching of more software engineering topics. Nevertheless, when developing online courses in software testing one should try to address the challenges of understanding the objectives and details of the testing topics explored in each assignment and how these testing assignments are supported by all the needed instructions and details. Some of the mitigations to the described challenges are already known to other domains [10,17]. This is an indication that these results could be relevant to other software engineering practices, both for an academic and an industrial perspective. For research, the experiences provided in this paper are useful as they bring knowledge from teaching industrial practitioners in an online setting into academia. By understanding that teaching practitioners is not a trivial approach, further research is made possible. We invite other researchers to revisit in other contexts the experienced challenges and good approaches.

Our main aim with this paper is to highlight the themes, challenges and approaches that arose from the development and running of online courses in software testing targeting industrial practitioners. This qualitative approach has produced an understanding of the challenges and good approaches in online teaching of software testing in the industrial context. This approach has its limitations regarding the small sample size of courses used in this analysis and the uni-dimensional collection of observations; therefore, no attempt to generalize the results of this study is made. We hope the findings of this study will stimulate further research on teaching software testing to industrial practitioners in an online setting.

6 Conclusions

This study has illustrated how advanced topics in software testing practice and research can be taught to industrial practitioners. We report several good approaches and challenges in developing and teaching distance web-based courses in three software testing topics: software testing theory of advanced test design, automated test generation and model-based testing. We identified several challenges in online teaching of software testing: poor feedback on practical assignments, distances between students and teachers, poorly supported assignments, and also some industrial practitioner-specific challenges. In addition, we identified several good approaches for improving online courses in software testing: active participation at the student's convenience and pace, inclusion of software artifacts from industrial practice and creation of short video lectures among others. Finally, our results show that more research on online teaching of software testing is needed and that teachers and researchers need to take the aspects of teaching industrial practitioners more clearly into account.

Acknowledgments

This research was supported by the The Knowledge Foundation (KKS) through the FuturE and PROMPT projects as well as the ECSEL Joint Undertaking under grant agreement No. 737494 and the Swedish Innovation Agency, Vinnova (MegaM@Rt2 and XIVT projects).

References

1. Ammann, P., Offutt, J.: Introduction to software testing. Cambridge University Press (2016)
2. Astigarraga, T., Dow, E.M., Lara, C., Prewitt, R., Ward, M.R.: The emerging role of software testing in curricula. In: 2010 IEEE Transforming Engineering Education: Creating Interdisciplinary Skills for Complex Global Environments. pp. 1–26. IEEE (2010)
3. Bazeley, P.: The contribution of computer software to integrating qualitative and quantitative data and analyses. Research in the Schools **13**(1), 64–74 (2006)
4. Biggs, J.B.: Teaching for quality learning at university: What the student does. McGraw-hill education (UK) (2011)
5. Boghossian, P.: Behaviorism, constructivism, and socratic pedagogy. Educational Philosophy and Theory **38**(6), 713–722 (2006)
6. Calderhead, J.: Teachers: Beliefs and knowledge. (1996)

7. De Freitas, S.I., Morgan, J., Gibson, D.: Will moocs transform learning and teaching in higher education? engagement and course retention in online learning provision. British Journal of Educational Technology **46**(3), 455–471 (2015)
8. Garousi, V., Rainer, A., Lauvås jr, P., Arcuri, A.: Software-testing education: A systematic literature mapping. Journal of Systems and Software p. 110570 (2020)
9. Garrison, D.R., Kanuka, H.: Blended learning: Uncovering its transformative potential in higher education. The internet and higher education **7**(2), 95–105 (2004)
10. Ghezzi, C., Mandrioli, D.: The challenges of software engineering education. In: International Conference on Software Engineering. pp. 115–127. Springer (2005)
11. Hadjerrouit, S.: Learner-centered web-based instruction in software engineering. IEEE Transactions on Education **48**(1), 99–104 (2005)
12. Hrastinski, S.: Nätbaserad utbildning: en introduktion. Studentlitteratur (2009)
13. Hynninen, T., Kasurinen, J., Knutas, A., Taipale, O.: Guidelines for software testing education objectives from industry practices with a constructive alignment approach. In: Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. pp. 278–283 (2018)
14. Karn, J., Cowling, A.J.: Using ethnographic methods to carry out human factors research in software engineering. Behavior research methods **38**(3), 495–503 (2006)
15. Miller, F.G., Fins, J.J., Bacchetta, M.D.: Clinical pragmatism: John dewey and clinical ethics. J. Contemp. Health L. & Pol'y **13**, 27 (1996)
16. Roth, W.M., Lee, S.: Scientific literacy as collective praxis. Public understanding of Science (2016)
17. Stuchlikova, L., Kosa, A.: Massive open online courses-challenges and solutions in engineering education. In: International Conference on Emerging eLearning Technologies and Applications). pp. 359–364. IEEE (2013)
18. Taguchi, N.: "contextually" speaking: A survey of pragmatic learning abroad, in class, and online. System **48**, 3–20 (2015)