On the Evaluation of an Open Software Engineering Course

Eduardo Figueiredo, Juliana Alves Pereira, Lucas Garcia, Luciana Lourdes
Software Engineering Lab (LabSoft) - Department of Computer Science (DCC)
Federal University of Minas Gerais (UFMG)
Belo Horizonte - MG, Brazil
{figueiredo, juliana.pereira, lucas.sg, lucianalourdes}@dcc.ufmg.br

***Abstract* — Open online courses are a method of online lecturing whose application in education is not bounded by space and location constraints. The successful implementation of open courses requires conceptual changes in how instructors and students behave in open unbounded education environment. There are some emerging open courses for teaching specific topics of Software Engineering. However, it is still limited the knowledge about the best practices for learning Software Engineering processes, methods, and tools in such an open environment. To address this limitation, this paper presents and evaluates an open course for Introduction to Software Engineering. The presented open course has over 250 online students registered and is based on a face-to-face equivalent. The online course is currently composed of 44 video lectures, 160 questions in 16 quizzes, and several discussion topics. We evaluate this course by comparing the students' performance in online vs. face-to-face equivalent courses. Our results indicated that students who had access to online content achieve similar or better performance than students taking only the face-to-face course.**

Keywords — open education; software engineering course; evaluation

I. INTRODUCTION

Internet has become an important tool to modify education through open online courses. An open online course (or open course, for short) creates an educational environment not bounded by space and location constraints [11]. The success of open courses requires conceptual changes in the way as lectures and students behave in such an open unbounded environment [11] [17]. Supporters of open education claim that the course content available is an important tool in the learning process. In some sorts of open courses, students can join the class at any time, do exercise, read past discussions, and so on [23]. Such innovation in terms of education allows discussion-oriented classrooms, i.e., face-to-face classes are dedicated for questions and activity solutions with closer interaction between students and teachers.

Several respectful universities around the world, mainly from North America and Europe, are providing open online courses based on their face-to-face equivalent courses. These courses have attracted great attention of hundreds of thousands of worldwide students [11] [23] [24]. For instance, Harvard University and Massachusetts Institute of Technology (MIT) have invested on the creation of various open courses by the edX portal [9]. In addition, more than 80 American and European universities, such as Princeton and Stanford, are involved with the creation of hundreds of open courses in several areas by means of Coursera [7]. Many other courses have been successfully created in Udacity [27] and similar open learning portals, such as Udemy [28]. In addition to video lectures, students regularly registered in an open course can participate of practical exercises and have an assessment of their performance. In some open courses, a student who successful completes the course obtains a certificate or statement of accomplishment.

There are a couple of online courses to teach specific subjects related to Software Engineering [23] [24] [26]. However, these courses focus either on specific advanced topics [23] [24] or on how to program [26]. For the best of our knowledge, there is no open online course to cover all basic subjects of a typical introductory course of software engineering. More important, there is no systematic study to investigate whether this way of teaching is efficient and viable to teach the basic concepts of Software Engineering, such as software process, requirements engineering, software maintenance, and so on. Therefore, it is essential for us to investigate and evaluate the actual benefits and drawbacks of open online courses in order to understand if and how such courses can indeed improve the learning of Software Engineering concepts.

Every year at the Federal University of Minas Gerais (UFMG) in Brazil, more than 100 students take a face-to-face Software Engineering course. Inspired by this fact, we created in 2013 an open online course for Introduction to Software Engineering. In this online course, students have access to 44 on-line video lectures and 16 quizzes. Moreover, conversations between students and the course instructors are possible due to the discussion forums and emails, which provide a highly interactive virtual community for Software Engineering learning.

We evaluated the open Software Engineering course by comparing the performance of students in three types of teaching method: face-to-face course, online course, and hybrid course; the latter includes features of both face-to-face and online courses. Students involved in this study are students taking the course for two consecutive years (2012 and 2013).

All students have the minimum requirements to enroll in the course and each student only took part in one teaching method. Our results indicated that (i) students taking a hybrid course achieved better performance than students taking only the face-to-face course, (ii) students taking a hybrid course achieved similar performance of students taking only the online course, and (iii) for both hybrid and face-to-face courses, frequency in lectures has a direct impact on the students' performance.

The remainder of this paper is organized as follows. Section II summarizes and discusses related work about open education in software engineering. Section III presents the structure of the open course, emphasizing its characteristics, students enrolled, and available content. Section IV evaluates the course and discusses the lessons learned. Section V discusses some of the study limitations and future work before concluding this paper.

II. Open Online Software Engineering Education

Open online courses have recently gained high popularity among various universities [26]. In these courses, students come from a variety of geographical locations and may range in the thousands to hundreds of thousands. Materials in open courses, such as recorded video lectures, readings, and assignments, are available via the Internet. Recently emerging learning platforms, such as Coursera [7], Udacity [27], and EdX [9], have been started to offer open courses in partnership with universities. Other learning platforms, such as Udemy [28], host open courses created by everyone, regardless of university partnerships.

Although these learning platforms provide substantial amount of courses to some discipline of Computer Science, such as programming and algorithms, they lack courses for a varied of Software Engineering specific topics, such as Requirements Engineering and Software Architecture. In fact, to the best of our knowledge, there are very few open courses for Software Engineering [21] [23] [24] and none of them covers basic Software Engineering topics. More important, there is no systematic assessment of these courses to verify whether they really support the learning of Software Engineering concepts.

For instance, Fox and Patterson [11] recently published a paper in the Communications of the ACM to discuss topics for online courses, including test-driven development (TDD) and agile software development. They report the use of agile development techniques to teach a face-to-face course with more than 100 junior and seniors, at Berkeley University. The authors [11] also discuss their experience in providing the first part as an open course to 50,000 online students, most of who work in the IT industry. However, the paper does not present any assessment to verify whether their open course has positive or negative aspects in the student learning. Similarly, Schmidt and McCormick [23] presented a course to teach design and architecture patterns for development of concurrent and networked software systems.

Both Fox and Patterson [11] and Schmidt and McCormick [23] papers focus on some specific topics in the long list of content covered by an introductory Software Engineering course. On the other hand, in this paper we present and evaluate an open course that teaches most of the extensive content of Software Engineering discipline, such as software process, requirements engineering, software design, and software maintenance.

III. A Software Engineering Open Course

This section describes an open course for introduction to Software Engineering. Subsection A summarizes the course structure. Subsection B presents the basic Software Engineering concepts covered by the course syllabus. Subsection C shows statistics of access to the course Web pages.

A. Course Summary

Two teachers have planned and delivered a face-to-face course for Introduction to Software Engineering at the Federal University of Minas Gerais (UFMG) in Brazil. This face-to-face course has already been delivered eight times following the same format from 2010 to 1012. In 2013, these two teachers and two teaching assistants, collectively called instructors in this paper, decided to create an online equivalent course. The goal of this online course is to support students to get easy access to the course material. As a result, we created and taught in 2013 a hybrid course for Introduction to Software Engineering inspired by our experience with the face-to-face course lectured in previous years. The hybrid course is composed of both online video lectures and face-to-face ones.

The online part of this course was hosted in the Udemy platform [1]. By using this platform, the course instructors made available video lectures, presentations, quizzes, and supplemental files. The course also allows students to participate and interact with instructors via discussion forums. In fact, students have a number of features to support online learning, in addition to traditional face-to-face classes. For instance, they can take notes during the video lessons and access to content using mobile devices.

The initial goal of the open course was to make its content available for students academically enrolled in the face-to-face equivalent course. Therefore, students could watch the missed classes or review the course material online. In this scenario, in addition to 4 hours of face-to-face classes per week, students would also have access to online content. However, with the progress of the course, some students - authorized by the respective teacher - have chosen not to attend face-to-face classes and just following the online course. Furthermore, due to the open and free course nature, several students registered to the online course, although they were not academically enrolled at the corresponding face-to-face classes.

As an introductory Software Engineering course, the open course we created covers the entire lifecycle of software development. The course describes, for instance, how to apply methods and techniques to deliver a higher quality product to the customer. The material used in the course (e.g., textbook, presentations, exercises, etc.) is the same for all students regardless if they are academically enrolled or just registered for the online course. However, students academically enrolled perform face-to-face exams while online students only do quizzes and exercises.

The current version of the course has 44 video lectures with over 20 hours of recorded content, 14 quizzes with 10 questions each, and several discussion topics in the forum. Each quiz has been answered by at least 20 students and the discussion topics have more than 400 postings in total. These numbers show not only the relevance of the available course content but also the engagement of students in the course. Currently, ten students rated the course in the Udemy portal; eight of them gave a 5 stars rating (the best evaluation) and two students gave 4 stars for the course in a 0 to 5 star scale.

Figure 1 presents information about students registered in the open online course, launched on March of 2013 as the academic year started at UFMG. The left-hand side of Figure 1 shows that most students are also enrolled in one of the undergraduate course at UFMG. The right-hand side indicates which undergraduate course students are enrolled in. About 56% of the registered students are also enrolled in the face-to-face course offered in UFMG. The reason for a relatively small number of students only registered for the online course is due to two main factors: (i) the course is new, i.e., it was created last year and (ii) the course has not been publicly advertised to worldwide students. Instructors have chosen to not publicly advertise the course in its first year for a greater control on the course evaluation (Section IV).

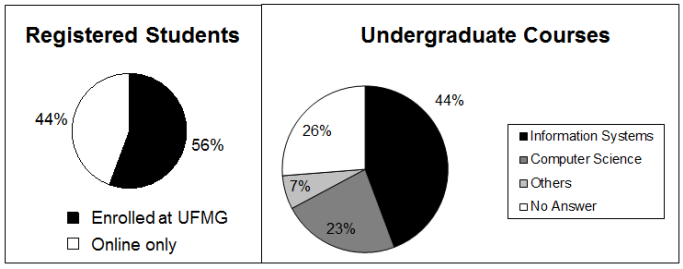

Fig. 1. Overview of students registered in the online course

Figure 1 also shows that students academically enrolled belong mainly to two undergraduate courses: Information Systems and Computer Science. About 70 % of students are taking one of these two courses. The explanation is because the Computer Science Department at UFMG offers these two options for undergraduate students. In addition, the corresponding Software Engineering face-to-face course is mandatory for these two curricula (Information Systems and Computer Science). The Software Engineering course is optional for other UFMG curricula.

B. The Course Content

Our Software Engineering course is mainly based on two text books: Software Engineering by Sommerville [25] and The UML User Guide by Booch, Rumbaugh, Jacobson [6]. The online version of this course is composed of 43 video lectures. Table I presents a list of the video lectures and maps each video lecture (2nd and 3rd columns) to the corresponding face-to-face lecture in the first column. Note that more than one video is mapped to a single face-to-face lecture because the former lasts up to 30 minutes and the latter is 2-hour long.

TABLE I. FACE-TO-FACE AND ONLINE COURSE CONTENT

Day	Video	Lecture Content
I	1	Course Presentation
	2	Introduction to Software Engineering
	3	Software Development and Evolution
II	4	Common Software Development Activities
	5	Software Process
	6	Software Process for Changing Software Systems
III	7	Agile Software Development
	8	Extreme Programming
	9	Scrum
IV	10	User Requirements
	11	Functional and Non-Functional Requirements
	12	Requirements Engineering
V	13	Introduction to Unified Modeling Language
	14	Use Case Diagrams
VI	15	Software Architecture
	16	Architecture Patterns
VII	17	Object Oriented Modeling
	18	Class Diagram
VIII	19	Sequence Diagram
	20	Using Sequence Diagrams to Detail Use Cases
	21	Collaboration Diagram
	22	Activity Diagram
IX	23	Object Oriented Programming
	24	OOP Idioms
X	25	Software Verification and Validation
	26	Software Inspection
	27	Software Testing
XI	28	Software Evolution
	29	Reengineering and Refactoring
XII	30	Introduction Software Reuse
	31	Software Reuse Techniques
	32	Software Product Lines
XIII	33	Component based Software Development
	34	Process for Component Developing
	35	Software Composition
XIV	36	Separation of Concerns
	37	Aspect Oriented Software Development
	38	The AspectJ Language
XV	39	Software Quality
	40	Measurement and Quality Models
	41	Software Product Metrics
XVI	42	Software Process Improvement
	43	Capability Maturity Model
	44	The Brazilian Software Process Improvement Model

We structured the open course into six sections as presented in Table II. This table also shows the mapping of sections to days, lectures, and quizzes. For instance, Section 1 of the course includes 6 lectures (i.e., lecture 4 to 9) and two quizzes. This section in the open online course has equivalent content to the one delivered in days II and III in the face-to-face Software Engineering course. We briefly describe below the content of each section of the course.

TABLE II. MAPPING OF SECTIONS TO COURSE LECTURES

Section	Days	Video Lectures	Quizzes
-	I	1-3	1
Section 1	II, III	4-9	2, 3
Section 2	IV, V	10-14	4, 5
Section 3	VI, VII, VIII	15-22	6, 7, 8
Section 4	IX, X, XI	23-29	9, 10, 11
Section 5	XII, XIII, XIV	30-38	12, 13, 14
Section 6	XV, XVI	39-44	15, 16

Before starting the first section, we added three introductory video lectures and one quiz that cover the background material needed to understand core motivation and concepts in the course. The introductory video lectures are about 40-minute long. The first video presents the course and the second one defines key concepts, such as software, software engineering, methods, and process. Similarly, the third lecture briefly explains the major activities in a software development process: requirements, design, implementation, testing, and evolution. Each quiz in the course has 10 questions.

As presented in Table II, Section 1 of the course contains 6 video lectures (about 2.5-hour content) and 2 quizzes. It discusses several software process models ranging from Waterfall [22] to the Spiral Model [5], and agile software development [4]. Section 2 includes five video lectures encompassing over 2 hours of record content about requirements engineering [15]. This course section also has two quizzes. The 3rd section contains three quizzes and about 4 hours of recorded videos spanned over 8 lectures. This section introduces the Unified Modeling Language (UML) [6] and outlines some of its main diagrams. It includes three quizzes with 10 questions each that allows students reviewing what they have learned.

Section 4 of the course contains about 4 hours of video lectures and three quizzes. The seven lectures present techniques for programming, testing, and evolving software code. Although there are examples of Java code in this course section, expertise in this specific programming language is not a pre-requirement to follow the lectures. Section 5 explains some techniques for software reuse, such as design patterns [12] [13] and software product lines [10]. This section is responsible for about 4 hours of video content and 3 quizzes. Finally, the last course section is composed of 6 video lectures (up to 3 hours in total) and 2 quizzes. It discusses various aspects of project management [18] [19], such as software quality, measurement, and software process improvement.

C. *The Course Audience*

We tracked accesses to the open course using the Google Analytics tool (http://www.google.com/analytics/). According to this tool, the online course had about 15,000 page views in the year of 2013; that is, more than one thousand visualizations of pages per month in average. The number is similar to what we observe in the current year (2014). For instance, Figure 2 (above) shows the number of page view in the first semester of 2014. The total number of page views from January to June of 2014 is 8,362, according to Google Analytics.

Figure 2 (below) also shows the statistics of visualization to the course pages in April of 2013; i.e., more than one year ago. We choose this month to illustrate a typical situation since the chart presents a similar shape in other months. From the data in Abril of 2013 (Figure 2 below), we observe that the visits were gradually increased from April 20th to 24th. In fact, students enrolled in the face-to-face course had an exam in 24th of April. Therefore, Figure 2 highlights that students used the online course material in order to prepare themselves for the exam. We observed similar behavior of students in every month with a face-to-face exam.

IV. Evaluation of the Open Online Course

The online education has brought new challenges and potential benefits to traditional education. This section aims to evaluate the benefits and drawbacks of online education based on the open course. That is, it discusses the results of a preliminary evaluation of the online course presented in Section III. We ground our observations on data from statistics collected from several sources, such as automated tools (e.g., Udemy and Google Analytics tools), face-to-face exams, and surveys. Subsection A presents the study settings and procedures we followed. Subsection B compares the performance of students in a traditional face-to-face course with students in a hybrid course. A hybrid course is a face-to-face course with additional online content (video lectures, quizzes, forums, etc.). Subsection C investigates the impact of

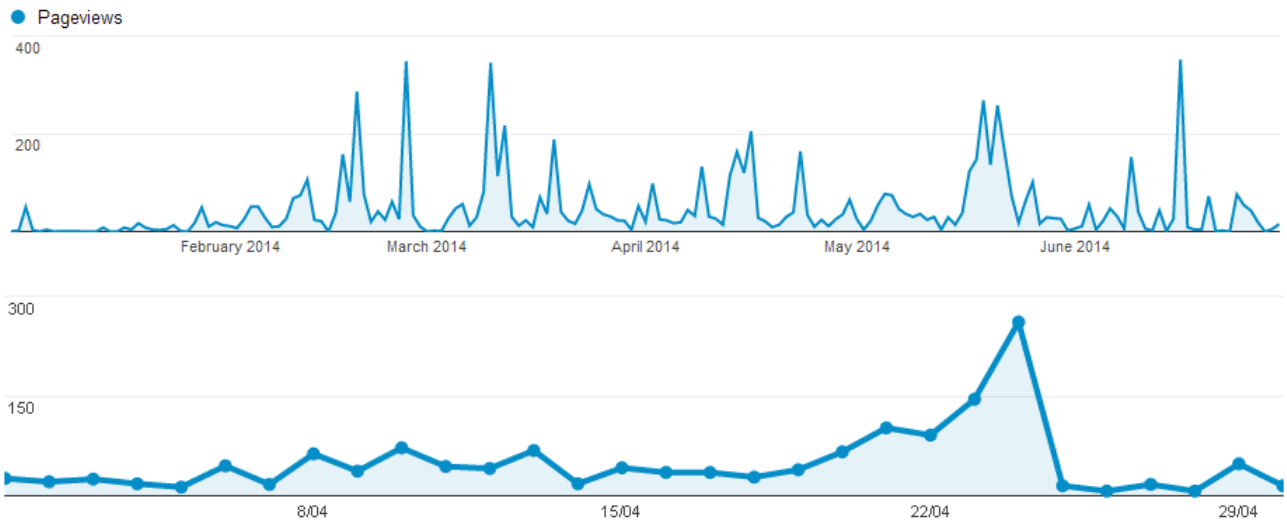

Fig. 2 Statistics of course page visualizations per day in January to June of 2014 (above) and in April of 2013 (below).

frequency of students in lectures on their performance. Finally, Subsection D evaluates whether purely online courses can replace face-to-face courses in Software Engineering without losing quality of learning.

A. Study Settings

This section presents the procedures we followed in this study. Our main goal is to identify the benefits and drawbacks of open online courses. Our general hypothesis is that online courses support face-to-face classes by enhancing student learning. Since we have not advertised the course to external students, students registered for the online course are mainly those regularly enrolled in the face-to-face course counterpart offered by the Department of Computer Science at UFMG. Students taking the course in 2013 both faced the same academic environment and were assessed by the same means of students taking the course in the previous years. However, in addition to face-to-face course, students in 2013 had support of the online content created and maintained by the instructors.

To evaluate the online course, we analyzed and compared the performance data from students in different academic years. In summary, we organize students in three groups according to the teaching methods: only face-to-face course in 2012, only online course in 2013, and hybrid course in 2013. In the case of 2013, we split those students who were regularly enrolled in the face-to-face course and those who were not.

- *Face-to-face Course*. Students at this teaching method had only attended a face-to-face course in 2012 (i.e., before we create the open online course). The final grade was based on two exams (60% of the final grade), one final project (30% of the final grade), and exercises in class or in a laboratory (10% of the final grade). The final project is often related to development, management, and maintenance tasks in a typical software system, such as Health Watcher [14] and MobileMedia [10].
- *Online Course*. Students at this teaching method are only registered for the online course (i.e., they were not enrolled at the face-to-face course). The course is fully available online for an unlimited number of students. Students are able to watch weekly video lectures in addition to take part in the discussion forum and answer quizzes. The online course follows the same general structure of the face-to-face counterpart, but students do not take exams.
- *Hybrid Course*. Students at this teaching method not only have face-to-face lectures but are also supported by online content. That is, students have the online video lectures available, for instance, when for some reason they cannot attend the face-to-face classes. Students in the hybrid course have access to the same resources of students in the online course. The final grade for the hybrid course is the same of the face-to-face course; i.e., two exams (60% of the final grade), one final project (30% of the final grade), and exercises (10% of the final grade).

B. Performance in Face-to-face vs. Hybrid Courses

In order to investigate the strengths and weaknesses of the online course, we compare the performance of students taking the face-to-face and hybrid courses. In this case, we conducted a comparative study involving 113 students taking these two kinds of courses; 56 students who took the face-to-face course in 2012 and 57 students who took the hybrid course in 2013. This section aims to answer the following research question defined.

RQ1. Does online content support learning of face-to-face courses?

In order to answer this question, we applied the t-test to analyze how better students in the hybrid course performed in comparison with students in the face-to-face course. The t-test [16] verifies whether the mean of two groups are significantly different. It uses different methods for paired and unpaired observations. In this step, we perform an analysis of unpaired observation, because we have independent samples from each of two populations. In other words, different groups of students were submitted to two different teaching methods (face-to-face and hybrid courses). In order to assist this comparison, two questions of the exams in 2012 and 2013 – called identical questions Q1 and Q2 – were the same. The remaining questions of the exams are not being considered in this analysis. Therefore, we ensure that the difficulty was exactly the same for all students in both groups (2012 and 2013).

Table III shows the summarization of students' grades for the two teaching methods: face-to-face and hybrid courses. The confidence level used for all analyzes was 90% (p-value = 0.1). As can be observed in Table III, the hybrid course has outperformed the face-to-face course. The confidence interval (CI) for the comparison of the face-to-face and hybrid courses does not include zero and, then, the positive mean difference indicates that the hybrid method is better. In other words, based on the means, we conclude that the grades of students submitted to the hybrid course is superior to students taking the face-to-face course. We conclude that when the subjects have access to a greater set of teaching tools, such as both the face-to-face and online courses, they are likely to obtain better grades compared with a single teaching method used in isolation. We may argue that a face-to-face course is not so

TABLE III. SUMMARY DATA FOR FACE-TO-FACE AND HYBRID COURSES.

Teaching Method	Face-to-face Course	Hybrid Course
x	7,288	8,706
S	2,048	1,703
X_{HYBRID}-$X_{FACE\text{-}TO\text{-}FACE}$	1,418	
S_{HYBRID}-$S_{FACE\text{-}TO\text{-}FACE}$	0,355	
v	109	
CI	(0.829; 2.007)	

efficient as a hybrid one because (i) the students are not freedom to choose where and when to study and (ii) when students miss classes, they skip content which reflects on lower grades.

C. The Impact of Frequency on Student Grades

This section aims to analyze whether the factor frequency of students has impact on their grades in a hybrid course. In other words, we aim to answer the following research question.

RQ2. What is the impact of frequency on the student grades?

To answer the above questions, we use the factorial design [16]. Factorial design is an important and simple experimental technique that enables the observation of the effects of the factors and their interaction. This is important to understand the impact of these factors on response variables and sorting out factors in the order of impact [16]. In this study, we are working with 22 experimental design [16]. The first evaluation step is to reduce the number of factors involved in the study and to choose those factors which may have significant impact on the grade of students. All students enrolled in the courses have at least basic knowledge in the relevant topics of Computer Science since the course has pre-requirements. Therefore, we decided to draw this analysis with respect to two factors: frequency and the teaching method.

We decide to focus our analysis on two levels of each of these two factors for the simulation. We then divided students in two categories according to their frequency: (i) infrequency indicates those students that missed more than 50% of lectures and (ii) frequent identify those students who show up in at least 50% of lectures. Additionally, we divided students in two categories according the used teaching method: (i) face-to-face course and (ii) hybrid course. In this analysis, we excluded students registered only in the online course since we have not assessed their performance in the course. That is, students of the online course do not take exams. This decision helps us decide if the difference of grades is significant enough to justify detailed examination.

Table IV shows the relation between the frequency of the students (infrequent or frequent) and the teaching method (face-to-face or hybrid course) provided to the analyzed students. Data presented in this table summarize the average grades obtained by students in two identical questions in 2012 (face-to-face course) and in 2013 (hybrid course).

TABLE IV. STUDENT GRADES IN TWO TEACHING METHODS

Frequency	Face-to-face Course	Hybrid Course
Infrequent	6,250	7,500
Frequent	7,413	9,016

Considering the impact of frequency and the teaching method on the grades of students, Table V shows the effects computed and the percentage of variance for each studied factor. The total variation is 3.860, of which 2,036 (53%) can be attributed to the teaching method, 1.794 (46%) can be attributed to frequency, and only 0.031 (1%) can be attributed to teaching method and frequency interaction (factor AB).

TABLE V. EFFECT FACTOR AND VARIATION EXPLAINED

Factor	Effect	Variation Explained (%)
Teaching Method	2,036	53%
Frequency	1,794	46%
Teaching Method and Frequency	0,031	1%
Total Variation	3,860	-

Based on data presented in Table V, we observed that the interaction study between the two factors of 1% variation seems irrelevant. The first factor, which explains 53% of the variation, deserves further analysis. In summary, we concluded that students without online support generally obtained lower grades compared to students who had the support of online course. Additionally, although less important than the teaching method, the variation of frequency is also high. That is, frequent students are able to take the best grades. Therefore, the frequency of students in hybrid courses is also important.

D. On the Need of Face-to-Face Lectures

This section aims to investigate if solo online courses are enough for learning Software Engineering concepts. In other words, our goal is to analyze whether the purely online course could replace traditional face-to-face courses. In this analysis, we compare the performance of students in a hybrid course with students following only the online course. We focus on answering the following research question.

RQ3. Con online courses replace traditional face-to-face courses?

To answer this question, we rely on two samples of students of the hybrid and online courses. Since students in the online course are not required to do exams, we select a group of 22 independent students to take two sections of the course and to answer the two equivalent questions. These students have not attended the face-to-face lectures and, therefore, they answer the questions based only on the online content. As with many real-world problems, in this case the population characteristics are unknown, and the goal of the analysis is to estimate these characteristics. For this evaluation, we applied the t-test [16]. Similarly to Section IV.B, the steps to determine the CI for the difference in mean performance requires us to summarize the results for its means, making an estimate of the variance, sample sizes, and an effective number of degrees of freedom presented in Table VI.

TABLE VI. SUMMARY OF DATA (HYBRID AND ONLINE COURSE)

Teaching Method	Hybrid Course	Online Course
x	4,828	5,199
S	2,278	2,725
X_{HYBRID}-$X_{ONLINE-COURSE}$	0,371	
S_{HYBRID}-$S_{ONLINE-COURSE}$	0,899	
v	17	
CI	(-1,194; 1,935)	

Table VI shows the summary of data for 90% confidence taking into account the two types of teaching methods. Based on these data, we observed that the difference between the two teaching methods in terms of grades is not relevant. This result

shows that students were able to recover similar grades, regardless of the teaching method (hybrid or online, in this case). This is an interesting result because it supports the claims that online courses are able to replace face-to-face or hybrid courses to some extent. The explanation could be due to the facilities of an online course that allows greater flexibility with content availability. For instance, students have access to all the content taught. Additionally, students can remove their doubts online before exams by watching the videos again, doing quizzes, revisiting discussion topics, and other resources available.

We believe that these results are justified by the fact that the Software Engineering discipline is quite expository. Therefore, online classes are excellent to see and review as often as necessary concepts of matter. Face-to-face courses, on the other hand, can be tiring due to its theoretical nature, making it difficult to assimilate the content. From the teacher perspective, face-to-face courses require greater availability of the teacher who teaches all classes in the course. However, after a substantial investment in creating an online course, the effort is paid off.

E. Threats to the Study Validity

The results of this study may be limited by a number of factors [29]. We made some decisions in order to mitigate the study limitations. For instance, when conducting a statistical evaluation, we have not use data from students who took the same course two times in different years, e.g., due to being reproved in the first try. As typical studies like this, other variables could not be fully controlled, such as the background and previous knowledge of students. We are aware of these limitations, but we believe that these factors do not invalidate our main conclusions. The results of this study is limited to the Software Engineering discipline, but we believe they may be replicable in other expository courses, such as in some other fields of engineering.

V. Conclusions and Future Work

This paper presents and evaluates an open online course for teaching introductory Software Engineering concepts. The online course is composed of 44 video lectures, 160 questions in 16 quizzes, and several discussion topics. In the evaluation, we directly compared the performance of students in a hybrid course with students in a face-to-face course (Section IV.B). We also compare the performance of students in a hybrid course with students in an online course (Section IV.D). However, we have not investigated whether the performance of students in online courses differs or outperforms the students in face-to-face courses. In other words, we do not have a final answer of whether the efforts and costs to maintain an online course are clearly justifiable if no face-to-face lecture is intended to be delivered. We plan to further investigate this issue in the future.

There are many differences between online and traditional face-to-face courses. For example, students in online courses had opportunity to watch classes and ask questions at any time and as many times as necessary. In addition, the quizzes allow instructors to gain better control of the students who need help in specific modules and may also help these students to get better results. However, results of this study represent only a first stepping-stone towards the understanding of the benefits and drawbacks of open online courses. As further work, we suggest that similar evaluation should be performed in other online courses.

Acknowledgment

This work was partially supported by FAPEMIG (Grant APQ-02532-12 and PPM-00382-14) and CNPq (Grant 485907/2013-5). We would like to thank all students who provided us with feedback about the online course.

References

[1] Open Software Engineering Course. Accessed in 08/07/2014: http://www.udemy.com/engenharia-de-software-ufmg/

[2] S. Apel, D. Batory, C. Kastner, G. Saake. Feature-Oriented Software Product Lines: Concepts and Implementation. Springer; 2013.

[3] M. A. Ardis and P. B. Henderson. “Software Engineering Education (SEEd) - Is Software Engineering Ready for MOOCs?”. ACM SIGSOFT Software Engineering Notes, v. 37, n. 5, 2012.

[4] K. Beck and C. Andres. Extreme Programming Explained: Embrace Change, 2nd Edition. Addison-Wesley, 2004.

[5] B. Boehm. “A Spiral Model of Software Development and Enhancement”, IEEE Computer, 21(5), 61-72, 1988.

[6] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide, 2 edition. Addison Wesley, 2005.

[7] Coursera. Accessed in 08/07/2014: http://www.coursera.org/

[8] E. Derwin. “Critical Thinking in Online vs. Face-to-Face Higher Education”. Media Psychology Review, vol. 2(1), 2009.

[9] edX. Accessed in 08/07/2014: http://www.edx.org/

[10] E. Figueiredo, N. Cacho, C. Sant’Anna, M. Monteiro, U. Kulesza, A. Garcia, S. Soares, F. Ferrari, S. Khan, F. Filho, and F. Dantas. “Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability”. In proceedings of the 30th International Conference on Software Engineering (ICSE), pp. 261-270. Leipzig, Germany, 2008.

[11] A. Fox and D. Patterson. “Crossing the Software Education Chasm”. Communications of the ACM, 55(5), 44–49, 2012.

[12] E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, 1995.

[13] A. Garcia, C. Sant'Anna, E. Figueiredo, U. Kulesza, C. Lucena, and A. von Staa. “Modularizing Design Patterns with Aspects: A Quantitative Study”. In proceedings of the 4th International Conference on Aspect Oriented Software Development (AOSD), pp. 3-14. Chicago, 2005.

[14] P. Greenwood, T. Bartolomei, E. Figueiredo, M. Dosea, A. Garcia, N. Cacho, C. Sant'Anna, S. Soares, P. Borba, U. Kulesza, and A. Rashid. “On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study”. In proceedings of the 21st European Conference on Object-Oriented Programming (ECOOP), pp. 176-200. Berlin, 2007.

[15] E. Hull, K. Jackson, J. Dick. Requirements Engineering, 3 edition. Springer, 2010.

[16] R. Jain. The Art of Computer System Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling, Wiley and Sons. 1991.

[17] T. R. Liyanagunawardena, A. A. Adams, and S. A. Williams. “MOOCs: A Systematic Study of the Published Literature 2008-2012”. The International Review of Research in Open and Distance Learning, 2012.

[18] M. Lorenz and J. Kidd. Object-Oriented Software Metrics, 1 edition. Prentice Hall, 1994.

[19] M. Lanza and R. Marinescu. Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer, 2006.

[20] K. Masters. “A Brief Guide to Understanding MOOCs”. The Internet Journal of Medical Education, 1, 2011.

[21] N. Papaspyrou, S. Retalis, S. Efremidis, G. Barlas, E. Skordalakis. “Web-based Teaching in Software Engineering”. Advances in Engineering Software 30, 901–906, 1999.

[22] W. Royce. “Managing the Development of Large Software Systems”, Proceedings of IEEE International Conference on Software Engineering (WESCON) 26, 1–9, 1970.

[23] D. C. Schmidt and Z. McCormick, “Producing and Delivering a Coursera MOOC on Pattern-Oriented Software Architecture for Concurrent and Networked Software”. In Proceedings of the International Conference on Systems, Programming, Languages and Applications (SPLASH). Indianapolis, 2013.

[24] Software as a Service (SaaS). http://www.edx.org/course/uc-berkeley/cs169-1x/software-service/691

[25] I. Sommerville. Software Engineering, 9 edition. Pearson, 2010.

[26] N. Tillmann, J. Halleux, T. Xie, S. Gulwani, J. Bishop. “Teaching and Learning Programming and Software Engineering via Interactive Gaming”. In Proceedings of the International Conference on Software Engineering (ICSE), pp. 1117-1126, 2013.

[27] Udacity. Accessed in 08/07/2014: http://www.udacity.com

[28] Udemy. Accessed in 08/07/2014: http://www.udemy.com/

[29] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, A. Wesslen. Experimentation in Software Engineering, 2nd edition. Springer, 2012.