Investigating How Features of Online Learning Support Software Process Education

Eduardo Fernandes, Johnatan Oliveira, Eduardo Figueiredo

Software Engineering Laboratory (LabSoft), Department of Computer Science (DCC)
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil

{eduardomorfernandes, johnatan-si, figueiredo}@dcc.ufmg.br

***Abstract*—Online courses are a method of lecturing whose application in education is not bounded by space and location constraints. They include features such as video lectures and online questionnaires. There are a few online courses to teach subjects related to Software Engineering. However, for the best of our knowledge, there is no online course to teach software process, which is a key area of Software Engineering. More important, there is no systematic study to investigate whether this way of teaching is efficient and viable to teach software process. This paper presents an empirical study to evaluate whether and how online features support the learning of software process in the light of an online Software Engineering course with 61 video lectures, 16 online questionnaires, and a discussion forum. This study relies on data of 100 undergraduate students over three consecutive years: 2014, 2015, and 2016. Data of this study suggest that students answer online questionnaires in order to review for face-to-face exams. Our results also show that videos and online questionnaires contribute to the improvement of up to 15% of student grades in software process questions when compared with students who neither watch videos nor answer online questionnaires. However, based on two exam questions that repeated over the three years, we verify that the grade improvement seems to be mostly related to video lectures watched, rather than to online questionnaires answered.**

Keywords—online education; software process; software engineering; empirical study

I. INTRODUCTION

Online courses are a method of lecturing whose application in education is not bounded by space and location constraints [25]. The success of open online courses requires conceptual changes in the way as lecturers and students behave in such an open unbounded environment. Supporters of open education claim that online features, such as video lectures and online questionnaires, are important tools to support learning [32].

In fact, several respectful universities around the world have been providing online courses based on their face-to-face equivalent courses. These courses have attracted great attention of hundreds of thousands of worldwide students [2]. As a result, many online courses have been successfully created and hosted in online education portals. Examples of such educational portals include Coursera[1], Edx[2], Udacity[3], and Udemy[4]. In some online courses, a student who successfully completes the course obtains a certificate or statement of accomplishment.

There are a few online courses to teach subjects related to Software Engineering. However, these courses focus either on software architecture, introduction to software engineering, or on how to program. For the best of our knowledge, there is no online course to teach general concepts of software process, including software process models, agile software development, and software process improvement. Software process is one of the knowledge areas to be covered in Computing undergraduate degree programs, such as Computer Science, Information Systems, and Software Engineering, according to the ACM and IEEE Computing Curricula [12].

In software engineering, a software process is a structured set of activities required to develop a software system [3]. To the best of our knowledge, there is no systematic study to investigate whether online courses are efficient and viable to teach software process. Therefore, it is essential for us to investigate and evaluate the actual benefits and drawbacks of online features, such as video lectures and online questionnaires, in order to understand whether and how such features can indeed improve the learning of software process concepts.

Since 2013, we have been used an educational online platform to support a face-to-face Software Engineering course [16]. The online features available for our students in this platform include 61 video lectures, 16 online questionnaires with 160 questions, and a discussion forum with almost one thousand posts. Around 15% of the course content (i.e., 10 videos and 3 online questionnaires) focuses on software process. In the light of this Software Engineering course, this paper presents an empirical study to evaluate whether and how online features, such as video lectures and online questionnaires, support the learning of software process.

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

[2] https://www.edx.org/

[3] https://www.udacity.com/

[4] https://www.udemy.com/

This study relies on data of 100 undergraduate students over three consecutive years from 2014 to 2016. We introduced the features of online learning gradually into the course; video lectures included in 2014 and online questionnaires in 2015. In all years of the course, students have to attend face-to-face lectures and do exams since they are regularly enrolled in an undergraduate degree program. In general, data of this study suggest that students watch video lectures and answer online questionnaires in order to review for face-to-face exams.

We also evaluate the grades of exam questions related to software process. Our results in this evaluation show that videos and online questionnaires contribute to the improvement of up to 15% of student grades in software process questions when compared with students who neither watch videos nor answer online questionnaires. However, based on two exam questions that repeated over the three years, we verify that the grade improvement seems to be mostly related to watched video lectures, rather than to answered online questionnaires.

The remainder of this paper is organized as follows. Section II provides background information to support the comprehension of the study. Section III describes the study design, including goals and research questions. Section IV presents our main research results. Section V discusses lessons learned. Section VI discusses some threats to the validity of our study. Section VII discusses related work. Finally, Section VIII concludes the paper and suggests future work.

II. BACKGROUND

This section provides background to support the comprehension of this study. Section II-A presents an overview of software processes. Section II-B discusses agile software development, including some examples of agile methods. Section II-C presents some relevant concepts of online education.

A. Software Processes

Software development is a complex activity that demands creativity and intelligence [31]. In software engineering, a software development process is composed of several tasks, methods, and practices to guide developers in the conception of a software product [24]. An effective software process has to consider clearly the relationship among different activities of the software development, generated artifacts, supporting tools, and required procedures. Moreover, software processes have to take into consideration training of development teams and personal motivation. The main purpose of software processes is to support development of high quality software products that fit the costumers' needs, through a well-defined schedule and planned budget [36].

In the context of software engineering, there is no perfect process, because the success of applying a software development process depends, for instance, on the settings of the development environment and the organization [31]. As an example, in the development of critical software systems as air traffic controllers (that demands well-designed and reliable implementation), it is required a strict software process to guide the software product development. In turn, for business software development that has short time-to-market deadlines and requirements that change constantly, a flexible and agile process may be more appropriate than other types of rigorous process [28].

Four activities are common in many software processes: Software Specification, Software Design and Implementation, Software Validation, and Software Evolution [31]. *Software Specification* consists of system functionalities and business rules defined for a software product. *Software Design and Implementation* is the development of software, in accordance with its specifications. *Software Validation* is the process of validating a software product to assure its quality and the accordance with the clients' needs defined as software specifications. *Software Evolution* is the activity of maintaining a software product in terms of features that may change along the time based on costumers needs.

Several software process models have been proposed in the software engineering literature. These models are generic and may be adapted depending on the development environment [35]. However, their overall definitions apply to general cases. We present some of these models as follows.

Waterfall Model. It consists of a linear model in which each development activity is conducted without iterations [20]. The main activities of the Waterfall model are requirements specification, development, validation, and evolution. The main advantages of the well-defined division for control and management of development phases [23]. Regarding drawbacks, this model is not flexible. That is, it does not consider reviewing the software artefacts after the activity has concluded. Another drawback is that the process can be difficult to adapt to requirement changes [8].

Incremental Development. This model is based on scheduling of activities related to specification, development, and validation [28]. The main goal of the Incremental model is to provide an evolutionary development of software products [21]. One of the advantages of this model is the iterative refinement of the software product, in accordance with its requirements [26]. However, one drawback of the Incremental Development is that the elaboration of robust software documentation may not be feasible [31].

Component-Based Software Engineering. In this model, the development of a software system is guided by the implementation of functional software components [19][31]. Each component is validated individually and, then, the integration of different components is evaluated [9]. The main goal of Component-Based Software Engineering is to support software reuse, since well-designed and validated components are usable in the composition of new software systems [13]. However, one of the main challenges in this software process is the development of flexible functional components with high quality [28].

We decide to deeper investigate software process in this study because it is one of the knowledge areas to be covered in Computing undergraduate degree programs, such as Computer Science, Information Systems, and Software Engineering [12]. In fact, software process is a key discipline not only for

Computing undergraduate programs but also to several related engineering courses. For instance, curricula of engineering programs, such as Electrical Engineering[5], also cover software processes as a relevant knowledge area.

B. Agile Software Development

Agile software development is a type of software processes that aims to increase time-to-market of software product delivery [11], as a response to the constraints imposed by traditional software processes, such as the Waterfall model [11]. The Agile Manifest[6] focuses on (i) individuals instead of processes and tools, (ii) functional software over extensive documentation, (iii) collaboration between stakeholders and development team, and (iv) response to changes. It compiles a set of 12 principles of agile software development, such as continuous software delivery, development team organization to provide appropriate software design, and attention to technical excellence of software product.

Fowler and Highsmith [17] present agile methods as a new development approach covering various programming practices. There are also many techniques to support the agile software development and some of them aim to provide high software quality and reliability [22], client satisfaction with respect to software product fast deliver [1], and abridged documentation for small development teams [10]. In general, agile software development techniques targets continuous refinement of software requirements, motivation of the development team, and different software quality aspects [17]. Some examples of agile methods are Extreme Programming (XP) [5], Scrum [6], Crystal Clear [10], and Test-Driven Development (TDD) [22].

Once agile software development is an important topic in software processes, both in academia and in industry, it has been taught in Software Engineering courses to support the formation of practitioners and researches. Moreover, agile software development is a topic closely related to the industrial context of software development because of its use of dynamic, interactive, and innovative methods. Therefore, we included this topic in classes covering software process. That is, our software engineering course [16] has online features, such as video lectures and questionnaires, targeting agile software development.

C. Online Features and Podcasts

Open learning platforms, such as Corsera and Udacity, provide substantial amount of courses to some discipline of Computer Science and Software Engineering, such as programming and algorithms [25] and software architecture [30]. However, to the best of our knowledge, there is no online course focusing on software process. More important, there is no systematic assessment of these courses to verify whether features of online courses really support the learning of software process concepts.

Online features have been used to support education of different study topics. Many of these features are also called podcasts. Podcast consists of the application of content availability in a streaming fashion, including audio, video, and text [18]. The use of podcasts to support learning may impact positively on the commitment of students with respect to classes, and increase their skills regarding a specific subject [33]. In general, podcast include theoretical content provided in online platforms, and then face-to-face classes may be dedicated to practical exercises conducted by the course instructor with students, to support assimilation of content. In this study, we investigate podcasts as a way to support learning of software process concepts.

III. Study Design

This section describes the configuration of our study. Section III-A presents the online Software Engineering course we created at the Udemy educational platform. Section III-B discusses the study goal and research questions to guide this study. Section III-C describes the participants of our study.

A. The Online Course at Udemy

In this study, we assess online features provided through an online course[7] created and hosted in Udemy, an educational online platform. We created this course in 2013 and, since then, we have over 500 students registered in the online course. It is an introductory course of Software Engineering for undergraduate students, aiming to support the face-to-face classes conducted in the Federal University of Minas Gerais (UFMG). This Software Engineering course is mainly based on two textbooks: Software Engineering by Sommerville [31] and The UML User Guide by Booch, Rumbaugh, and Jacobson [7]. We structured the proposed open course into six sections: Software Process, Requirements Engineering, Software Design, Development Techniques, Software Reuse, and Software Quality.

The online features of this course include 61 video lectures (with a total of 20 hours), 16 online questionnaires (160 questions), and a discussion forum for each topic of study. The questionnaires cover all topics of the video lectures. From the overall online content available for students, there are 10 video lectures and 3 online questionnaires covering software process. That is, online features target at three main topics: general concepts of software processes, agile software development, and software process improvement.

B. Goal and Research Questions

In this study, we aim to investigate how features of online learning may support software process education in the context of a software engineering course. For this purpose, we have used of the course created in the Udemy platform over three consecutive years: 2014, 2015, and 2016.

Based on the Goal-Question-Metric (GQM) method [34], we summarize our study goal as follows: *Analyze how features of online learning support software process education, from the purpose of assessing the effectiveness of these features in student performance, with respect to correctness in course*

[5] https://www.ece.utexas.edu/undergraduate/courses/360f

[6] http://agilemanifesto.org/

[7] https://www.udemy.com/engenharia-de-software-ufmg/

exams, from the view of point of educators, in the context of Brazilian students.

To guide our study and, consequently, to support our data analysis, we designed three research questions as discussed below.

RQ1. *Do video lectures impact the performance of undergraduate students with respect to exam grades in software process?*

The purpose of RQ1 is to guide the evaluation of whether students who watch video lectures are able to reach higher grades in exam questions regarding software processes compared to students who do not watch video lectures.

RQ2. *Do online questionnaires impact the performance of undergraduate students with respect to exam grades in software process?*

In turn, with RQ2 we are concerned about the impact of students answering online questionnaires on their grades in exam questions related to software process.

RQ3. *Do online-based classes provide the same student performance than hybrid classes?*

Finally, with RQ3, we aim to compare two different learning methods, namely online-based and hybrid classes. Students in online-based classes are registered for the online course, but they do not have face-to-face classes about software process. On the other hand, Students in hybrid classes not only have face-to-face lectures but are also supported by online content. We assess the impact of using online features in grades of students in these two methods of learning.

C. Participants

Table I presents the set of participants, considering the classes from three consecutive years (2014, 2015, and 2016). The participants compose the dataset we aim to evaluate in this study. We have a total number of 100 undergraduate students under analysis, after we discard of data from 9 students that eventually quit the course.

TABLE I. PARTICIPANT SET

Participants	**Year**			**Sum**
	2014	**2015**	**2016**	
Before Reduction	20	33	56	109
Removed Participants	2	3	4	9
After Reduction	18	30	52	100

Figure 1 illustrates the final distribution of participants per year. This figure make it clear that about half of students took the course in 2016. In fact, the number of students has grown year after year. This observation may suggest that the course popularity has increased and, so, it has attracted more students in each new offer.

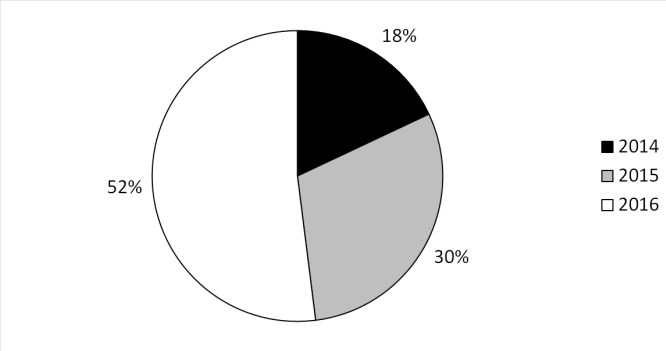

Fig. 1. Distribution of participants per class

Note that classes in 2014 and 2015 used a hybrid approach for learning, by combining online features (in this case, video lectures and online questionnaires) and face-to-face classes. In 2016, we decide to have just online classes (with the same video lectures and questionnaires) specifically for the topics of software processes and agile software development. Therefore, data from 2014 and 2015 refer to hybrid classes in both years. However, the online course in 2014 includes only video lectures (no questionnaire about software process). For this reason, we first pairwise compare 2014 (hybrid, but no questionnaire) with 2016 (online only) and then we pairwise compare 2015(hybrid) with 2016 (online only).

IV. RESULTS

This section presents the results of our study and provides some discussion regarding the support of video lectures and online questionnaires in the learning of software processes. Section IV-A evaluates classes from two consecutive years, namely 2014 and 2015, with respect to the impact of video lectures and online questionnaires on the performance of students. Section IV-B discusses a comparative study, considering two pairs of classes: 2014 with 2016 and 2015 with 2016. The purpose in this case is to investigate the impact of online features on leaning, without considering the support of face-to-face classes.

A. Video Lectures and Online Questionnaires

We conducted a first study to evaluate data from two consecutive years (2014 and 2015) with respect to the support provided by video lectures and online questionnaires in the software process learning. Therefore, we investigate the research questions RQ1 and RQ2, considering the participant set described in Section III-C. Let us focus first on the first research question (RQ1).

RQ1. *Do video lectures impact the performance of undergraduate students with respect to exam grades in software process?*

One common question appears in exams of both years: 2014 and 2015. Then, we analyzed this question to assess how the score achieved by participants in this question is correlated with the frequency of watched video lectures and the answers of online questionnaires. The context of these questions is about agile software development.

With respect to video lectures, we divided students in three groups (low, regular, and high frequency) based on the distribution of percentage of watched videos. We computed the 33% and 66% percentiles to divide students in groups with similar sizes. Based on these percentiles, we obtained the following thresholds: 4.6 hours (for 33%) and 8.4 hours (for 66%). These values mean, for instance, that 66% of the participants watched up to 8.4 hours of the 20 course video hours. We considered all videos from the online course because Udemy does not provide the number of watched videos per topic. Although this overall analysis of videos do not cover only videos related to software processes, we assume that a student that watches videos with high frequency probably has also watched videos from the specific topic of software process.

Table II presents the number of participants ("# Participants") from each group. "VID" is the percentage of watched video lectures where 0.23 means 4.6 hours and 0.42 means 8.4 hours. "Mean Score" represents the average of score that participants achieve for the exam question. Note that, the higher is the frequency of watched videos, the higher is the achieved score in the exam question. For instance, students with low frequency of watching videos obtained a mean of 0.85, while students with high frequency of watching videos achieved a mean of 1 (full score). Therefore, we conclude that the video lectures available for students support their learning of software processes.

TABLE II. FREQUENCY OF WATCHED VIDEO LECTURES VERSUS MEAN QUESTION SCORE

Frequency	Threshold	# Participants	Mean Score
Low	VID < 0.23	13	0.85
Regular	0.23 <= VID <= 0.42	19	0.89
High	VID > 0.42	16	1.00

Comparing the Low and High frequency groups, we observe and increase of 15% in the mean of scores in question. We also computed the Pearson correlation coefficient between the percentage of watched videos and the score achieved by participant in the exam question. We obtained a 0.23 Pearson correlation which is considered as a "weak" correlation. Therefore, we conclude that the positive impact of watching video lectures on software processes learning does not seem to be strong.

RQ2. *Do online questionnaires impact the performance of undergraduate students with respect to exam grades in software process?*

Regarding online questionnaires, we did not divide participants by score in questionnaires because Udemy provides the correct answers in the end of questionnaire filling. The platform also allows students to redo the questionnaire many times if they want. Therefore, the score obtained by participants is biased and inappropriate for evaluation. Then, we considered only whether participants completed the questionnaires. We then compute correlation between their scores in the exam question and the completion of the questionnaire.

We computed the Pearson correlation coefficient between the completion of online questionnaires and scores in the exam question. The obtained coefficient is 0.18, classified as "very weak" correlation. Therefore, we assume that questionnaires have less impact the software process learning than watching video lectures. However, this online feature may be used as complementary to face-to-face classes, as well as video lectures, since both features have a positive impact on students' grade.

B. Hybrid versus Online-based Learning

This study compares two different learning methods for software processes learning (see Section III.C). First, we have hybrid classes (in 2014 and 2015) combining face-to-face classes with online features (video lectures and online questionnaires). Second, we have one class (2016) with only online support for software processes learning. In this case, we aim to assess whether online features provides the same benefits for students, in terms of scores in exam questions, when compared with the hybrid learning methods. Therefore, we investigate RQ3 (below). Note that, in this evaluation we do not asses the frequency of watched video lectures, because this data is not available for student from 2016 since the course has not been concluded when this paper was written. Therefore, we compare only scores in exam and completion of questionnaires.

RQ3*. Do online-based classes provide the same student performance than hybrid classes?*

We identified two common questions that appears in exams from the classes of 2014 and 2016. Then, we selected these questions to assess how the score achieved by participants in these questions is correlated with the learning methods (hybrid or online-based). The context of both questions is about agile software development. In turn, with respect to exams from 2015 and 2016, we identified only one common question (the same used for evaluation in Section IV-A). Then, we selected this question for the comparison between hybrid and online-based learning approaches. In this evaluation, we also did not divide participants by score in questionnaires due to the same reasons discussed in Section IV-A.

We conducted a two-step comparative analysis. First, we compare 2014 and 2016 classes to assess the benefits of using a hybrid method when compared with online-based classes. Table III presents the scores in question with the number of completed questionnaires. The first and second columns of this table indicate the year and learning method, respectively.

"Mean SE" is the mean score for the two exam questions under analysis. "Mean CQ" is the mean rate of completed questionnaires.

TABLE III. COMPARISON BETWEEN THE YEARS OF 2014 AND 2016

Year	Method	Mean SE	Mean CQ
2014	Hybrid	0.92	1.00
2016	Online	0.84	0.88

Based on data in Table III, we observed that the hybrid class (2014) presented a slightly higher mean of scores in the exam questions when compared with the online-based class (2016). In this case, we observe an increase of 8% in the mean of score. Therefore, we conclude that video lectures may support learning of software processes without replacing the face-to-face classes. Moreover, online questionnaires do not seem to impact significantly in the student performance.

We performed a similar analysis to compare performance of students in 2015 and 2016. Table IV shows the scores in the exam question with the number of completed questionnaires. Table IV has a similar structure to Table III. That is, "Mean SE" is the mean score for the exam question and "Mean CQ" is the mean rate of completed questionnaires.

TABLE IV. COMPARISON BETWEEN THE YEARS OF 2015 AND 2016

Class	Method	Mean SE	Mean CQ
2015	Hybrid	0.90	0.87
2016	Online	0.87	0.88

Again, we observed that the hybrid class (2015) presented a slightly higher mean of scores in exam question when compared with the online-based class (2016). In this case, students from the 2015 class obtained a mean of 0.9 for the exam question, and the students from the 2016 class achieved a mean of 0.87 (a lower difference when comparing 2014 and 2016). In other words, we observed an increase of 3% in 2015 when compared with 2016. Therefore, we reinforce that video lectures support learning of software process, but with no replacement to the face-to-face classes. Considering online questionnaires, we observed the same rate of completed activities for both classes and, so, they do not seem to impact on the students' performance.

V. LESSONS LEARNED

Through this study, we are able to observe some interesting findings with respect to the online features we applied in class. First, as stated in Table II, 13 out the 48 participants (around 27%) presented low frequency of watched videos. This fraction of participants is significant, mainly because all participants are required to watch videos during the course. Since the videos of our course, available at Udemy, have a mean of 16 minutes of length, we may consider shortening the videos to provide more focused videos. With this action, we hope that students may be motivated to watch more videos.

Second, we have observed a very weak correlation between students' score in exam question and the completion of online questionnaires (see the discussion regarding RQ2). This observation leads us to draw some treatment to improve the quality of these questionnaires. At Udemy, we provide only 3 online questionnaires for 10 videos; that is, about three videos for each questionnaire. Therefore, we consider the creation of new questionnaires to cover topics of each video in near future. The goal would be to improve the effectiveness of online questions in the online learning.

VI. THREATS TO VALIDITY

We carefully designed and conducted our study as described in Section III. For instance, we delimited the study scope prior to the data analysis, defined our research questions, and how to assess each of them, based on previous and related work. However, there are some threats to validity that may impact our research findings. Following, we discuss each of the four types of threats, with respective treatments, as proposed by Wohlin et al. [34]: internal, conclusion, construct, and external validity.

Construct Validity. In this study, we aim to evaluate whether online educational features, such as videos and questionnaires, support the student performance in software processes learning. For this purpose, we designed a study to evaluate effectively our research questions. We provided access to the online features for every participant. To provide impartiality of students with respect to our research questions, we omitted this information to all students. Therefore, we expected a minimization of biases with respect to the impact of online features on the student performance.

Internal Validity. The supporting materials of face-to-face classes, as well as the video lectures, have been prepared by the course instructor and may change along the years. In general, these modifications aim to improve the quality of teaching resources, and may include rearrangement of study topics, explanation approach, and other aspects. However, to minimize this problem, the data we chose for analysis in this study corresponds to unchanged content only. Another threat to internal validity is that engagement of students in online activities is uncontrolled variable. That means the use of online features by students is not manageable, mainly because of factors such as personal motivation and eventual distractions. This lack of control may have affected the quality of the learning process we assess in this study. To minimize this problem, we rewarded participants with points in class grade for each complete online activity.

Conclusion Validity. This study investigates the use of online features in three classes (from three consecutive years) with different number of students. All classes were placed in the same educational institution and conducted by the same instructor. In this context, the application of our study in different classes may minimize threats regarding diversity and representativeness of the participant set. We carefully selected data related to software processes learning, i.e., exam questions, online questionnaires, and face-to-face classes were

considered for data analysis. We based our data collection selection on related work, to ensure that such data would be useful in drawing conclusions. We conducted a careful data analysis to draw study conclusions, to minimize problems with respect to data interpretation. We also chose carefully descriptive analysis techniques to present results appropriately.

External Validity. This study focus on a Software Engineering course. Therefore, our results are related to this specific research area and may not be generalized to other disciplines. Moreover, we assessed data only from undergraduate students from UFMG. Therefore, we may not assure our conclusions are applicable in other educational institutions. However, the classes we evaluated are composed by diversified participants, and this variety may minimize problems with the participant set representativeness.

VII. Related Work

The adoption of new methods to support the education processes may be motivated by many reasons, such as to motivate students and focus on their learning process. In this context, many methodologies have been proposed in literature, such as educational games [4] and multidisciplinary environments [14][29]. In general, these approaches aim to support students in the comprehension of learning topics.

With respect to courses related to Computer Science, many methods have been proposed to support students with respect to knowledge acquisition. For instance, we may find studies that propose interactive tools for specific topics such as Jeliot [27]. This tool aims to help students from early stages of programming courses to learn the object-oriented programming (OOP) paradigm. Jeliot simulates to students, in a visual fashion, the running of OOP programs.

In the context of Software Engineering courses, many studies have been investigating the use of different methods to support superior education. For instance, Day and Foley [15] investigate the use of online features to improve learning in computing-related courses. They conduct a quasi-experiment with student of a Human-Computer Interaction course at Georgia Tech. One group of students had their face-to-face classes replaced by in-class practical exercises and the content of the course was provided exclusively online. The online content included 27 classes in video (a total of 9 hours of videos) to support learning of the course subjects. The other group developed their learning activities in a traditional way, with face-to-face classes only. As a result, the authors observed an improvement of students' motivation in the online-based approach, and this motivation provided better grades for these students when compared with students from the class based only in face-to-face learning.

Another example of study is provided by Gannod et al. [18]. The authors investigate the application of a hybrid educational approach that combines face-to-face and online education. This hybrid method was applied in a Software Engineering from University of Miami. The chosen subject for analysis is Special Topics in Service-Oriented Architecture. The experiment design consists of 65 online context (podcasts) provided for all students. During the face-to-face classes, the course instructor answered questions regarding the online content, and then practical exercises were conducted. As a result, authors observed that: (i) all students reported that podcasts are significantly useful in the learning process and provide a space for discussion and practical activities in face-to-face classes, (ii) 92% of the students agree that podcasts are not sufficient to support learning, and (iii) 56% of the students agree that podcasts may be used as a complement for face-to-face classes without replacement to this format of class.

In previous work, Figueiredo et al. [16] describe another empirical study regarding a hybrid approach for educational support an introductory Software Engineering course from Federal University of Minas Gerais (UFMG). We present an open online course in the Udemy online platform, with around 180 enrolled students, 44 video lectures, 140 questions for topics covered by the videos, 14 online questionnaires, and discussion forums for each topic of study. Through a comparison of students' performance, we assess: (i) face-to-face classes, online features, and the proposed hybrid educational approach composed by online features and face-to-face classes. The proposed work relies on data of students from two consecutive years: 2012 and 2013. As a result, we observe that students enrolled in the hybrid course had better performance and grades in course when compared with students form the method based only on face-to-face classes. Moreover, students from the hybrid classes performed similarly when compared with students from strictly online-based classes. We also observe that online features may support face-to-face classes without replacing this class format.

In this paper, we take our previous work [16] as basis to investigate the learning support of online features in the context of software processes, a specific Software Engineering topic. This study relies on the analysis of two features: video lectures and online questionnaires. We also analyze data from three consecutive years: 2014 to 2016. The main study goal is to assess how these online features impact the students' performance in terms of grades in course.

VIII. Conclusion

This paper presents an empirical study to evaluate the application of online features as a support for learning of Software Engineering courses. Specifically, we focus on the analysis of two different online features (video lectures and online questionnaires) and specific subject topics: software process and agile software development. Our study relies on data from 70 undergraduate students over three consecutive years, from 2014 to 2016.

We divided our study in two parts. First, we gather data from 2014 and 2015, with respect to a common exam question regarding agile software development, to assess the correlation between scores achieved by participants in the exam question and the frequency of watched video lectures per participant. Second, we pairwise compare the student performance in the years 2014-2016 and 2015-2016 to evaluate how different is the score of students in hybrid classes (composed by online features and face-to-face classes) with respect to only online-based classes.

In general, we observed that video lectures and online questionnaires do not replace face-to-face classes provided by instructors. In addition, we assume that the hybrid approach (composed by face-to-face interactions and the support of online features) provides better student performance when compared with only online-based classes. Therefore, our study points that online features are useful as complementary methods to the traditional format of class, with slightly positive impact on the scores in exam of participants.

As future work, we suggest the evaluation of other educational supporting features, such as discussion forums, games, and practical exercises, to investigate whether support learning of software processes. Other suggestions is investigate the use the same educational institutions. We also suggest a study to investigate the impact of applying online features considering the overall student performance in the superior education of Software Engineering courses. For instance, we may assess the impact of using online features in other topics than software processes and agile software development.

ACKNOWLEDGEMENTS

This work was partially supported by CAPES, CNPq (grant 485907/2013-5), and FAPEMIG (grant PPM-00382-14).

REFERENCES

[1] S. Ambler and M. Lines, *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press, 2012.

[2] A. Amresh, A. Carberry, and J. Femiani, "Evaluating the Effectiveness of Flipped Classrooms for Teaching CS1," in *Proceedings of the 43th Frontiers in Education Conference (FIE)*, 2013, pp. 733–735.

[3] M. Aoyama, "Agile Software Process and Its Experience," in *Proceedings of the 20th International Conference on Software Engineering (ICSE)*, 1998, pp. 3–12.

[4] A. Baker, E. Navarro, and A. Van Der Hoek, "An Experimental Card Game for Teaching Software Engineering Processes," *Journal of Systems and Software (JSS)*, vol. 75, no. 1–2, pp. 3–16, 2005.

[5] K. Beck, *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2000.

[6] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2007, pp. 255–264.

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

[8] B. Boehm, "A Spiral Model of Software Development and Enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, 1988.

[9] X. Cai, M. Lyu, K.-F. Wong, and R. Ko, "Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes," in *Proceedings of the 7th Asia-Pacific Software Engineering Conference (APSEC)*, 2000, pp. 372–379.

[10] A. Cockburn, Crystal Clear: *A Human-powered Methodology for Small Teams*. Pearson Education, 2004.

[11] D. Cohen, M. Lindvall, and P. Costa, "An Introduction to Agile Methods," *Advances in Computers*, vol. 62, pp. 1–66, 2004.

[12] *Computing Curricula 2005: The Overview Report*. ACM and IEEE Computer Society, 30 September 2005.

[13] I. Crnkovic, "Component-Based Software Engineering for Embedded Systems," in *Proceedings of the 27th International Conference on Software Engineering (ICSE)*, 2005, pp. 712–713.

[14] D. Damian, A. Hadwin, and B. Al-Ani, "Instructional Design and Assessment Strategies for Teaching Global Software Development: A Framework," in *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, 2006, pp. 685–690.

[15] J. Day and J. Foley, "Evaluating a Web Lecture Intervention in a Human-Computer Interaction Course," *IEEE Transactions on Education*, vol. 49, no. 4, pp. 420–431, 2006.

[16] E. Figueiredo, J. Pereira, L. Garcia, and L. Lourdes, "On the Evaluation of an Open Software Engineering Course," in *Proceedings of the 44th Frontiers in Education Conference (FIE)*, 2014, pp. 1–8.

[17] M. Fowler and J. Highsmith, "The Agile Manifesto," *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[18] G. Gannod, J. Burge, and M. Helmick, "Using the Inverted Classroom to Teach Software Engineering," in *Proceedings of the 30th International Conference on Software Engineering (ICSE)*, 2008, pp. 777–786.

[19] G. Heineman and W. Councill, *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Professional, 2001.

[20] M. Huo, J. Verner, L. Zhu, and M. Babar, "Software Quality and Agile Methods," in *Proceedings of the 28th International Computer Software and Applications Conference (COMPSAC)*, 2004, pp. 520–525.

[21] I. Jacobson, G. Booch, J. Rumbaugh, J. Rumbaugh, and G. Booch, *The Unified Software Development Process*. Addison-Wesley, 1999.

[22] D. Janzen and H. Saiedian, "Does Test-Driven Development Really Improve Software Design Quality?" *IEEE Software*, vol. 25, no. 2, pp. 77–84, 2008.

[23] D. Kirk, S. MacDonell, and E. Tempero, "Modeling Software Processes: A Focus on Objectives," in *Proceedings of the 24th Conference Companion on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, 2009, pp. 941–948.

[24] M. Komuro, "Experiences of Applying SPC Techniques to Software Development Processes," in *Proceedings of the 28th International Conference on Software Engineering (ICSE)*, 2006, pp. 577–584.

[25] K. Masters. "A Brief Guide to Understanding MOOCs". *The Internet Journal of Medical Education*, 1(2), 2011.

[26] P. Mohagheghi, B. Anda, and R. Conradi, "Effort Estimation of Use Cases for Incremental Large-Scale Software Development," in *Proceedings of the 27th International Conference on Software Engineering (ICSE)*, 2005, pp. 303–311.

[27] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, "Visualizing programs with jeliot 3," in *Proceedings of the 6th Working Conference on Advanced Visual Interfaces (AVI)*, 2004, pp. 373–376.

[28] R. Pressman, *Software Engineering: A Practitioner's Approach*, 5th ed. McGraw-Hill Higher Education, 2001.

[29] V. Razmov and R. Anderson, "Pedagogical Techniques Supported by the Use of Student Devices in Teaching Software Engineering," *ACM SIGCSE Bulletin,* vol. 38, no. 1, pp. 344–348, 2006.

[30] 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)*, 2013.

[31] I. Sommerville, *Software Engineering*. Pearson Addison Wesley, 2010.

[32] J. Strayer, "How Learning in an Inverted Classroom Influences Cooperation, Innovation and Task Orientation," *Learning Environments Research*, 2012.

[33] C. Wellington, J. Lee, S. Joseph, and L. Kennedy, "Work in Progress: Podcasts for Recruiting and Retaining Female Computer Science Majors," in *Proceedings of the 37th Frontiers in Education Conference (FIE)*, 2007, pp. S2H–1–S2H–2.

[34] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

[35] H. Zhang, R. Jeffery, D. Houston, L. Huang, and L. Zhu, "Impact of Process Simulation on Software Practice: An Initial Report," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, 2011, pp. 1046–1056.

[36] K. Zielinski and T. Szmuc, *Software Engineering: Evolution and Emerging Technologies*. IOS Press, 2005.