Massive Open Online Courses in Software Engineering Education

Aracele Garcia de Oliveira Fassbinder
Federal Institute of Education, Science and Technology of South of Minas Gerais, Muzambinho, Minas Gerais, Brazil
aracele.garcia@ifsuldeminas.edu.br

Marcelo Fassbinder
University Cruzeiro do Sul
Guaxupé, Minas Gerais, Brazil
marcello.fassbinder@gmail.com

Ellen Francine Barbosa
University of São Paulo (USP)
São Carlos, São Paulo, Brazil
francine@icmc.usp.br

George D. Magoulas
Birkbeck College, University of London
London, United Kingdom
gmagoulas@dcs.bbk.ac.uk

***Abstract*— Despite the popularity of MOOCs in providing opportunities for socialization, collaboration, and professional improvement, there has been little research exploring them in the context of Software Engineering Education (SEE). The purpose of this study is to provide a better understanding of practices and challenges when developing academic software engineering MOOCs. To this end, we research (i) how MOOCs in SEE are designed and (ii) what learning design considerations educators and learning designers have when developing SEE-related MOOCs. To address the first research question, a systematic mapping is performed to outline the current landscape. The second research question is answered by analyzing and reflecting on practical experiences acquired during the design of a MOOC on Agile Software Development. The design of the course was based on the Pedagogical Design Pattern Framework for MOOCs, an approach to design for learning that was defined in our previous studies. For this course, two environments were considered: (a) course-time on the Tim Tec MOOC platform for deeper engagement with active learning activities, such as Project-based learning; (b) a Facebook group as a social and collaborative learning space, including scenarios based on problem-based learning to activate prior knowledge. Some of the pedagogical strategies adopted to motivate self-regulated learning included self-introduction video, diary, diagnostic self-assessments, development of small projects alone and in pairs. Our findings provide evidence that there are still several technological and pedagogical challenges that need to be addressed so as to enhance the MOOCs learning experience for SEE. Technological issues identified include the development of tools and educational games, which should be integrated into MOOCs, and the use of learning analytics to support motivation, user experience, and more active learning in specific topics, such as Project Management, Agile Methods, and Requirements Engineering. In turn, the most significant and needed improvement to the pedagogical aspects is a re-thinking of the virtual moment in the MOOC platform so as to optimize it through activities that promote active learning and contextualized assessments.**

Keywords—MOOCs; Software Engineering Education; Agile Software Development; Lifelong Learning; Learning Design; Design Patterns.

I. INTRODUCTION

Collaboration is at the core of software engineering both in industry and in academia. In industry, most projects, apart from trivial ones, are inherently cooperative, requiring many software designers and programmers to coordinate their efforts to produce a large software system. In academia, software engineering is most of the time taught through collaborative activities, where teams of trainees take different roles and collaborate, either on-site or remotely, to solve a challenging task. Although Massive Open Online Courses (MOOCs) offer opportunities and tools for socialization, collaboration, and professional improvement [7, 8], their potential in Software Engineering Education (SEE) remains under investigated [15]. Figueiredo [9], for instance, states that "there is still limited the knowledge about the best practices for learning Software Engineering processes, methods, and tools in such an open environment.". García-Penalvo and Colomo Palacios [16] also highlight that "the central question is how to achieve the personalized interaction that Engineering processes require with the massive audience of MOOC approaches".

To alleviate this situation and improve our understanding of SEE-related MOOCs, the paper investigates the following research questions:

RQ1: How are SEE-related MOOCs designed?
RQ2: What are educator's considerations with respect to design for learning in SEE-related MOOCs?

Thus, following section, Section II, describes a mapping of MOOCs. Section III outlines the design of a MOOC on Agile Software Development. Section IV details and discusses the results of the mapping and the MOOC development to derive a set of lessons learned about the implications of designing MOOCs for software engineering education. The paper ends with the conclusion and ideas for future works.

II. MOOCS FOR SEE: A MAPPING

To answer RQ1, "How are MOOCs in SEE designed?", a systematic mapping of MOOCs was performed to understand the practice in the design for learning in this context. The mapping is divided into two steps: (i) data

978-1-5090-5920-1/17/$31.00 ©2017 IEEE

collection and (ii) an initial explorative study to understand how software engineering-based MOOCs are pedagogically designed. Partial results are described for each step, and general mapping results are discussed last.

A. Data Collection

Only MOOCs categorized in Computer Science or Software Engineering areas were considered. Data from a variety of sources were collected between December 2016 and February 2017. To avoid bias, a protocol was defined. Data were collected by one specialist and reviewed by another one.

First, the following MOOC providers were used to obtain software engineering-related virtual open courses: Coursera (www.coursera.org), MiríadaX (miriadax.net), Udacity (udacity.com), and edX (www.edx.org). These providers require quality control procedures to be followed by any institution and educator who want to deliver open courses using their platforms. Besides, they have been defined as leading providers by specialized information media [3, 4]. For each course, the following data were collected: course name, web address, main subject, duration in weeks, language, fundamental requirements, dedication time required, and the name of the university offering the course, among others.

Second, two websites to search about MOOCs were used to complete the information identified previously and to build a full dataset to be investigated further. They are also named aggregator sites: MOOC List (www.mooc-list.com) and Class Central (www.class-central.com).

Then, the identified courses were organized in subareas according to the topics defined in the Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [5]. The guidelines draw upon a range of sources, such as the Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, the Conference on Software Engineering Education & Training (CSEE&T), and the Guide to the Software Engineering Body of Knowledge (SWEBOK), among others. It also includes advances in teaching technologies, such as MOOCs.

B. Intermediate analysis (step 1)

The previous step led us to identify 87 MOOCs. Most of them were delivered in English (67%), followed by Portuguese (20%), Spanish (12%), and French (1%).

Table 1 presents a distribution of MOOCs and the average of required hours of study per week, by MOOC provider.

TABLE 1 - DISTRIBUTION OF MOOCS BY PROVIDER.

Provider	Number of identified MOOCs	Average of required hours of study per week
Coursera	18	5
MiriadaX	9	5.52
Udacity	10	5.3
edX	50	6.47

Table 2 outlines a general distribution of MOOCs by software engineering main subareas, as defined in [5].

TABLE 2 – MOOCS DISTRIBUTION BY SE SUBAREAS.

Subareas	MOOCs
I.Computing Essentials: Computer Science Foundations; Construction technologies and tools.	64
II.Mathematical and Engineering Fundamentals: Mathematical foundations; Engineering foundations for software; Engineering Economics for software.	3
III.Professional Practice: Group dynamics and psychology; Communications skills (specific to SE); Professionalism.	3
IV.Software Modeling and Analysis: Modeling foundations; Types of Models; Analysis fundamentals.	1
V.Requirements Analysis and Specification: Requirements fundamentals; Eliciting Requirements; Requirements specification and documentation; Requirements validation.	2
VI.Software Design: Design concepts; Design strategies; Architectural design; Human-computer interaction design; Detailed design; Design evaluation.	1
VII.Software Verification and Validation: V&V terminology and foundations; Reviews and static analysis; Testing; Problem analysis and reporting.	5
VIII.Software Process: Process concepts; Process implementation; Project planning and tracking; Software configuration management; Evolution processes and activities.	6
IX.Software Quality: Software quality concepts and culture; Process Assurance; Product Assurance.	1
X.Security: Security fundamentals; Computer and network security; Developing secure software.	1

C. Explorative study

To understand how software engineering-based MOOCs have been pedagogically designed, we performed an initial explorative study involving the courses identified in the subareas (iv) Software Modeling and Analysis and (v) Requirements Analysis and Specification. Such instances were selected considering their theoretical nature and the difficult usually found by students to learn software engineering foundations [16].

Their learning designs were captured and analyzed considering the main ideas defined in [12]. Firstly, we looked at (i) "What are the activity types commonly used in the context of open and online software engineering-based MOOCs", and (ii) "How many are the hours estimated for each type of activity undertaken?". Secondly, related data were collected by a learning design specialist and reviewed by a different one to improve information reliability. For each module in each MOOC in the sample, the following data were collected: type of activity and the number of hours that students are expected to study in each activity, which is very complicated to measure once students can start, re-start and finish their education journey at any time. The learning activities considered were respectively described and used in Rienties and Toetenel [12] and Fassbinder et al. [6], and are summarized in Table 3.

TABLE 3 – TAXONOMY OF LEARNING ACTIVITIES ACCORDING TO [12, 6].

	Type of activity	Example
Assimilative	Attending to information.	Read, watch, listen, think about, access, observe, review, study.
Finding and handling information	Searching for and processing information.	List, analyze, collate, plot, find, discover, access, use, gather, order, classify, manipulate.
Communication	Discussing module related content with	Communicate, debate, discuss, argue, share,

	another person (student/ tutor).	report, collaborate, present, describe, question.
Productive	Actively constructing an artifact.	Create, build, make, design, construct, contribute, complete, produce, write, draw, refine, compose, synthesize, remix.
Experiential	Applying learning in a real-world setting.	Practice, apply, mimic, experience, explore, investigate, perform, engage.
Interactive/Adaptive	Applying learning in a simulated setting.	Explore, experiment, trial, improve, model, simulate.
Assessment	All forms of assessment (diagnostic, summative, formative, and self-assessment)	Write, present, report, demonstrate, critique.

D. Intermediate analysis (step II)

As a result from the capture of the learning designs, Fig. 1 can be used to visualize general tacit pedagogical decisions made by MOOC instructors. The figure provides some insight on common patterns identified in the way educators have been designing open and online courses in the context of software engineering.

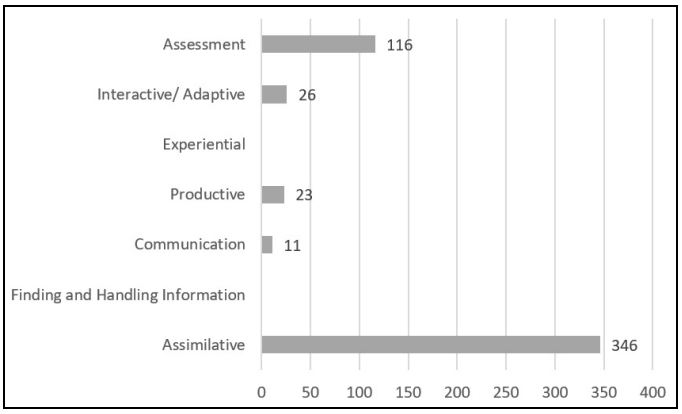

FIGURE 1 – AVERAGE OF HOURS THAT STUDENTS ARE EXPECTED TO SPEND PER TYPE OF ACTIVITY IN SEE-RELATED MOOCS.

E. Current results from the mapping of SE-based MOOCs

The mapping of MOOCs and the preliminary analysis of their learning designs provided first insights and evidence to answer RQ1, "How are SEE-related MOOCs designed?". The analysis revealed that there is a concentration of MOOCs especially about "Computing Essentials", which includes computer science foundations to support software product design, construction, and knowledge about the transformation of a design into an implementation, and techniques/tools used during this process [5]. Also, there is a tendency to not create multi-language courses when the first language is not English. For example, MOOCs in Portuguese are available only in this language, and this could be a limitation considering the 'openness' characteristic behind the concept of MOOCs.

Furthermore, software engineering MOOCs are traditionally taught deductively, in general. There are videos where the instructor introduces a topic by lecturing on general principles, then forums and quizzes to test student´s ability to answer questions considering the content explained in videos. Fig. 1 highlights educators' preferences of using more assimilative and assessment activities in their learning designs. The average of hours that students are expected to perform the activities is also high. Moreover, regular assimilative activities include reading texts and watching videos, and frequent assessment activities involve multiple question choice.

III. A CASE ABOUT AGILE SOFTWARE DEVELOPMENT

In addition to mapping the landscape of MOOCs' design in the context of software engineering education, and to answer RQ2: "What are educator's considerations with respect to design for learning in SEE-related MOOCs?", a MOOC on Agile Software Development was delivered. Section III(A) outlines the main motivation; Section III(B) discusses the Pedagogical Design Pattern Framework used to design the MOOC; Section III(C) presents the MOOC development itself; and Section III(D) reports data analysis, statistics, and findings from delivering the MOOC.

A. Motivation

Despite the maturation and development of the software engineering area and the agile movement, the use of agile methods in software engineering face-to-face or virtual courses is still a relatively new phenomenon [1]. According to Melo [2], adjustments in curricula on software engineering education to conform to this new paradigm of software development are necessary. However, this is a challenging endeavor that requires alignment between "theory, hands-on experiences, real projects, time to develop software, a place for building the agile environment, and the presence of several roles and stakeholders" [2]. We believe that this issue deserves further investigation, so we decided to focus on a MOOC about agile development to support lifelong learning.

Also, a face-to-face and fee-paying training course that inspired our Agile Software Development MOOC has some limitations. Training has been offered periodically by an Information Technology small business based in the south of Minas Gerais, the countryside of Brazil. Due to space limitation, only 12 people can attend the course each time it is delivered. The course also requires specific prerequisites of technical nature, such as experience in Ruby on Rails (i.e. web-application framework), Git (i.e. version control system) and Docker (i.e. software container platform). In contrast, our MOOC is free of cost, virtual, open to everyone with experience in at least one programming language (e.g. Java, C, Python, among others). Activities and learning materials have been adapted to include people with different levels of backgrounds and interests in Agile Software Development. Nevertheless, making innovative changes from a face-to-face context to a MOOC also poses several challenges. The performed adaptations and design of the new course are described next.

B. Pedagogical Design Pattern Framework: An Overview

In previous works, we defined a conceptual framework to support the design of learning experiences for MOOCs in Computer Science [6]. The framework brings together important features and principles that provide a deeper understanding of MOOCs in this area and related subjects. It also provides a systematic support for educators and learning

designers when developing learning maps for open and virtual courses. Additionally, the framework can also be used to support research in this context.

The framework is based on theoretical principles from three main subareas (i) characteristics and requirements for Massive Open Online Courses; (ii) Flipped Learning and active teaching-learning methodologies as a pedagogical strategy [18, 6]; and (iii) Pedagogical Design Patterns and Pattern Language [19, 6] as a strategy to record and share tacit design knowledge.

In general, the framework consists of (1) a life cycle, which summarizes fundamental steps to plan, deliver, and evaluate a MOOC; (2) a pedagogical design pattern language for MOOCs, which is based on problems and recurring solutions to solve the main activities described in the MOOCs lifecycle; and (3) related artifacts. They are briefly described next.

1) A life cycle for MOOCs

Fig. 2 summarizes a generic Life Cycle for MOOCs.

Exploration	Planning	Development	Offering	Update
• Framing the idea (e.g. initial content, team, learning objectives). • Identifying essential features of a MOOC project. • Analysing the MOOC platform. • Developing a collaborative syllabus (mapping the context, user stories gathering)	• Preparing the Learning Map and related artifacts (e.g. video scripts, user characterization survey) using: ✓ Pedagogical Model for MOOCs inspired in Flipped Learning ideas and principles. ✓ Mini-models of active learning strategies adapted for MOOCs. ✓ Assessment and monitoring strategies of learning. • Using a instrument of validation of learning maps in the MOOC context.	• Recording and editing videos. • Preparing other learning objects (e.g. readings, games, simulators). • Transferring the Learning Map to the choosen MOOC Platform (e.g. learning activities, objects, surveys, videos).	• Delivering the MOOC using the choosed Platform. • Monitoring and feedback.	• Analisyng findings from the MOOC final evaluation. • Identificatio n of future improvements and adaptations.
Evaluation: Learning Map evaluation; Models of Assessment; MOOC final evaluation.				

FIGURE 2 – A LIFE CYCLE FOR MOOCS [6].

In the "exploration phase", MOOC teams need to understand the teaching and learning challenges by diving into the context and the target audience of the course. User stories can be gathered to understand what the intended audience wants to learn, how and why. It is also important to identify university/institutional needs to define business and delivery models. In the "planning phase", the collected stories will help the MOOC team to set the syllabus and design the MOOC pedagogical model. It means, for instance, choosing the main pedagogic structure used to identify interdisciplinary opportunities, setting learning goals, and developing a Learning Map. A Learning map contains learning activities and is part of a learning design process. In general, it contains attributes such as the course name, general and specific learning goals, units, learning objects and activities to be performed by students. During the "designing phase", the stories selected in the Planning phase are implemented using some MOOC provider (e.g. Coursera, edX, MiríadaX, among others), or a MOOC open platform (i.e. Tim Tec, Google Course Builder, and open edX). Finally, the "offering or delivering phase" consists in running the MOOC. Evaluation activities can be performed during all phases. The MOOC can be tested using quality assurance methods formulated by specialists and considering perceptions from the audience [6].

2) A pedagogical design pattern language for MOOCs

Considering the activities described in the MOOCs lifecycle and the recurring solutions to implement them, a Pedagogical Design Pattern Language for MOOCs was built [6]. A pedagogical design pattern describes a recurring problem or challenge, the characteristics of the context in which it occurs, and possible solutions. Patterns are organized into coherent systems called pattern languages. Patterns are related to each other and thus offer a toolkit of interrelated design solutions that can be applied to novel problems [19, 20]. Fig. 3 provides an overview of the approach – interested readers are referred to [6] for details.

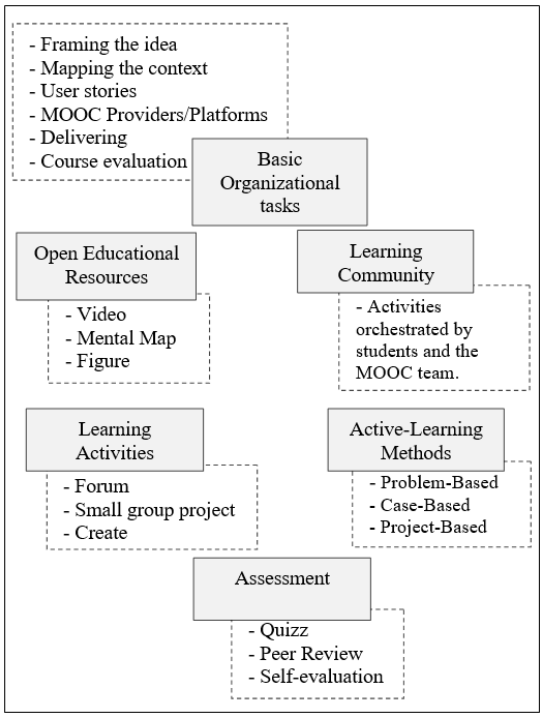

FIGURE 3 - A PEDAGOGICAL DESIGN PATTERN LANGUAGE FOR MOOCS [6].

3) Related Artifacts

The following related artifacts seek to support specific activities in the development of MOOCs considering the life cycle and the Pedagogical Design Pattern Language: (i) a model of Learning Map; (ii) an instrument to validate learning maps in the MOOC context; (iii) and a model of self-assessment and self-reflection instrument for students who achieved all the learning outcomes of the course. For this paper, an instrument to validate learning maps in the MOOC context [10] will be shortly described because the lessons learned presented in Section 4 are organized considering its fundamentals.

The artifact is a reliable and validated guide for MOOC teams. It contains 35 indicators grouped along eight dimensions. Table 4 presents the main dimensions, which represent desirable characteristics that should be considered by instructors when designing a Learning Map in the context of open and online courses. The associated indicators have been omitted due to lack of space.

TABLE 4 – DIMENSIONS OF DESIRED CHARACTERISTICS FOR MOOC LEARNING MAPS [10].

Dimensions
(1) Competency-based design
(2) User-centered learning
(3) Self-Regulation Learning
(4) Social Networking and Collaborative Learning
(5) Deep Learning, knowledge creation and sharing
(6) Assessment and Understanding
(7) Accommodate differences
(8) Feedback

C. Development

In this section, we describe the design of our MOOC on Agile Software Development MOOC organized along the main phases defined in the Life Cycle (Fig. 2).

1) Exploration

The following pattern was used to collect user stories from the target public: *As (function, job title or user profile), I (should, want to, would like to) (action or goal), with the purpose of (value for personal or professional life).* A mind map was used to resume and organize the main needs, curiosities, suggestions, characteristics, and content appointed by the target audience. To verify the current design of related MOOCs, the study presented in Section II (Fig. 1) was also considered.

Our MOOC was created and delivered using an instance based on Tim Tec MOOC Platform (http://timtec.com.br/pt/). Tim Tec is a Brazilian open source MOOC platform that can be utilized and personalized by any person or institution. In Tim Tec, classes are the central components of courses and they are responsible for the content. Classes are composed of units. Each unit requires a video. Activities are optional in the units. In its current version (v3.2), there is no support for peer assessment or internationalization. The course was complemented by a Facebook group as a collaborative learning space. Portuguese was the main language to develop videos, texts, and other educational resources.

2) Planning

Considering the main syllabus and Tim Tec available resources, we used the Pedagogical Design Pattern Language (Fig. 3) to develop the learning map related to the Agile Software Development MOOC. Project-Based Learning was used as the main active learning strategy to guide activities in the platform. Problem-Based Learning was used to guide discussions in the MOOC's Facebook group. The map's main topics and activities are summarized in Table 5. Learning goals were defined by considering Bloom's Taxonomy [13].

TABLE 5 - LEARNING MAP OVERVIEW.

MOOC: Agile Software Development Platform: mooc.ifsuldeminas.edu.br (Tim Tec)		
Goals	***MOOC Platform***	***Facebook group***
Before starting the MOOC		
Remember Understand Create	How the course works; Personal presentation through recording and sharing video interview; Initial question about challenges when developing systems; Self-evaluation.	Sharing the video interview. Sharing opinion about challenges when developing systems;
Welcome Class		
Remember Understand Create	Welcome video by instructors; Survey and characterization of students; Guidelines for Glossary-Based Collaboration; Readings.	Using #PairWithMe to find a partner.
Traditional Approach to Software Development		
Remember Analyze Create Evaluate	Project 1: Seven days to develop some functionalities, using student's current knowledge.	Sharing problems and new issues that hamper software development.
Manifesto for Agile Software Development		
Understand Analyze	Video interviews with software engineers about their contact with the Agile Manifest, and its values and principles.	Problem-solving using values and principles defined in the Agile Manifesto.
Iterative and incremental development		
Understand Analyze Apply Create Evaluate	Project 2: 7 days to 3 fast deliveries (3 days, 2 days, 2 days).	Sharing problems and new issues that still hamper software development. Making comparisons between traditional and agile development approach.
Agile practices: A big list		
Understand Analyze Apply Create Evaluate	Hands-on, interview videos, and practical pair activities about Kanban, Scrum, XP, Pair Programming, Virtual Pair Programming, Test-driven development.	Sharing experiences when programming in pairs.
Consolidation of Experiences		
Remember Analyze Evaluate	Webinar; Final self-evaluation; Last updated in Learning Mosaic.	Questions for a webinar with Software Engineers that use Agile strategies;

3) Developing

In this phase, specific videos were recorded and edited using our university media service. Videos were designed considering patterns of best practices when producing multimedia online videos for MOOCs, defined in the Pedagogical Design Pattern Language for MOOCs. Content in each week consisted of approximately five or six small videos from one up to five minutes of duration. All the learning materials used in the course, such as videos, images, and texts, were also based on common Open Educational Resources (OERs) guidelines [14]. Finally, the Learning Map (Table 5) was implemented in Tim Tec.

4) Delivering

The course was launched in November 2016 and stayed online for five weeks. When the course opened, all lessons were released. Although students were encouraged to follow the prescribed timeline of activities, they were free to complete it at their own pace. The first week was called "Before starting the MOOC". It was reserved to introduce some initial questions to motivate registered users to reflect on concepts to be worked during the course, and to regulate their learning process.

5) Course Evaluation

Data to verify the MOOC's effectiveness were collected from questionnaires, reports from the MOOC platform, and a semi-structured interview with a sample of 20 students for deeper clarification of some points.

D. Statistics and Data Analysis

About 600 students enrolled in the MOOC. Considering the university's preliminary investment on this type of learning modality, this demand was high. The current average number of enrollments per course on our institution' MOOCs is around 50 users. From a student's survey, we identified that at least three reasons contributed to the higher enrollment rate: greater interaction with market professionals, use of active learning strategies (project and problem-based learning), and increasing demand for professionals with knowledge in agile practices.

The geographic reach of the course was also significant. Students from 20 different Brazilian states (Brazil has a total of 26 states and one federal district, both distributed in 5 regions) registered on the course. Fig. 4 illustrates the distribution of students by state.

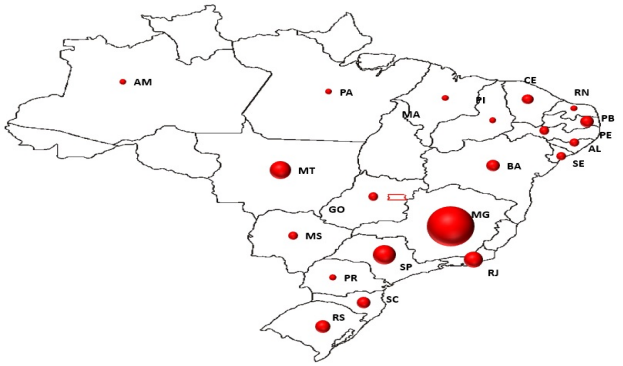

FIGURE 4 - DISTRIBUTION OF STUDENTS BY BRAZILIAN STATE. OUR INSTITUTION IS SITUATED IN THE STATE NAMED MINAS GERAIS (MG).

It is worth noticing the distribution of students by degree summarized in Table 6.

TABLE 6 - DISTRIBUTION OF STUDENTS BY DEGREE (LOWEST TO HIGHEST).

Current highest degree	% of Students
High school in progress	2.56
High school	4.27
Bachelor degree in progress	44.44
Bachelor degree	21.37
Specialization in progress	0.85
Specialization	9.40
Master degree in progress	2.56
Master degree	7.69
Ph.D. in progress	4.27
Ph.D.	2.56

Around 12.7% of those enrolled finished the MOOC having completed more than 70% of the proposed activities. Although being a small value, we need to highlight that the current average completion rate for our institution' MOOCs is around 5%. Besides, according to Liyanagunawardena [17], defining 'dropout' in MOOCs is a difficult task because this term is traditionally used in face-to-face and formal educational contexts. In MOOCs, participants can join a course just to follow a particular topic or to explore it. Once their goals are achieved, they voluntarily withdraw.

Taking into account interventions made in the MOOC Learning Map, we found that around 62.5% of students who completed the MOOC stated that deadlines suggested by instructors were not considered. This situation indicates that although suggesting dates and deadlines to complete activities by is a strategy to support self-regulation, the dates should be flexible for those who want to follow their own learning path and pace of study.

Close to 59.4% attended activities in all environments, both MOOC platform and Facebook group; 37.5% used only the MOOC platform; and 3.1% only the Facebook group. Although several other digital tools can be used to complement learning, such as blogs and social networks, this is evidence that MOOC platforms remain the primary environment. In this case, the Facebook group proved to have less influence on student's motivation to conclude the course.

When asked about their knowledge level in programming languages, most participants said they had some knowledge in C and Java; and no knowledge of Python and Ruby at all. The knowledge distribution seems uniform when verifying student's skills to develop complete programs with PHP, CSS, JavaScript, and databases (Fig. 5). There has been evidence that the lack of advanced or intermediate knowledge in programming languages contributed to the dropout rate in this case. It is important to highlight that the minimum requirement to subscribe the MOOC was basic programming knowledge in at least one programming language.

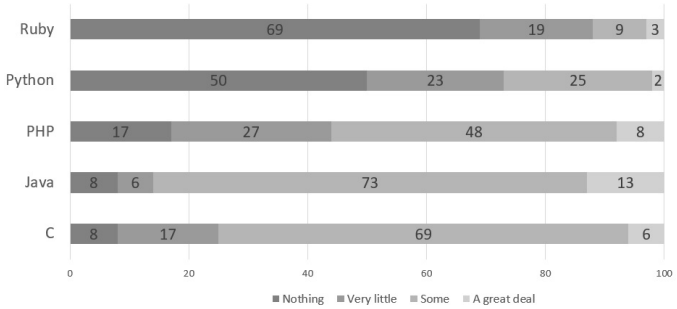

FIGURE 5 – PROGRAMMING LANGUAGE KNOWLEDGE DISTRIBUTION (IN %)

When asked about the main differences between the Agile Software Development MOOC and other online courses that the related students had already completed, the main areas identified were: interaction between theory and practice; the methodology used (project and problem-based learning); our MOOC put the focus on students and not on instructors; schedule flexibility; and more opportunities for interaction and collaboration between peers. Thus, there are indications that the interventions in the MOOC pedagogical approach were perceived and they were well-received by students.

Students were also asked about primary main intention when enrolling in the course: "I want to do all the activities to complete the course and reach a certificate" (71.9%), "I want to explore the course, see how it works, but I'm not sure if I'll complete it" (28.1%).

At the end of the course, we also asked if their initial intentions were fulfilled. About 32% said "yes", they did everything they had planned to do in the course. The others indicated problems that prevented them from achieving prior goals, such as problems to find someone to work as a pair when this was required by some programming activities; difficulty in organizing time and creating a study schedule to carry out the MOOC activities successfully.

Almost 40% of the students preferred working independently to working in pairs. Some of them reported that stopped doing the activities in pairs because of the difficulty to find an adequate mate.

In general, students initial learning goals were achieved, considering this distribution: yes, totally (12.5%); yes, a lot (68.8%); yes, partially (14.1%); yes, slightly (4.7%). Thus, there is evidence that students' expectations were met by the course. Students who said that their learning goals were partially or slightly achieved, attributed that to their difficulty in finding an active peer to work with on the pair programming activities and the lack of advanced knowledge in programming to easily perform coding activities.

We also used the characteristics for MOOC Learning Maps presented in Table 4 to identify student's opinions about learning strategies used to develop the MOOC. Specific subquestions (indicators) were used to collect data about each particular characteristic, as exemplified next.

TABLE 7 – EXAMPLE OF SUBQUESTIONS RELATED TO TABLE 4.

Characteristic (i.e. dimension): Competency-based Design
Subquestion 1: I liked the course used problems and projects as a teaching-learning strategy.
Subquestion 2: I liked the course contemplated a learning process more oriented to the accomplishment of activities than to the exposition and theoretical explanation of content (lectures).

The Five-point Likert scale was used to establish the level of student's agreement. Each level was denoted as: 5, I totally agree; 4, Agree; 3, Neither agree nor disagree; 2, Disagree; and 1, Totally disagree. In general, student's considerations about the MOOC learning design were positive. In this paper, due to lack of space, two extremes were selected and analyzed: one very positive and another implying some indifference:

- Prevalence of "Totally Agree":
  - I liked the course used short videos (1 to 5 minutes), to guide students in activities that should be developed (Competency-based design).
  - I enjoyed taking the course at my own pace (User-Centered Learning).
  - I felt motivated to continue learning and discussing agile practices even after the course is over (Deep-Learning, creation, and sharing of knowledge).
  - I liked the course used a variety of media to capture and retain my attention (Accommodate differences).
  - Messages sent by instructors were helpful (Feedback).
  - I liked the interview videos with software engineers (Competency-Based Design).
- Uniform distribution, with the prevalence of "Neither agree nor disagree":
  - Time reserved for pre-start activities was relevant to my learning process (Self-Regulation Learning).
  - I liked the strategy of recording a video interview for presentation (Social Networking and Collaborative Learning).

From these answers, it was possible to notice some relevant points: the importance of a closer relationship between academic and market professionals, and use of strategies to support mainly communication, self-expression, metacognition, and self-regulation skills.

As presented in the Learning Map (Table 5), students were asked to find relevant terms related to Agile Software Development and put them in a glossary. This web activity was also used to promote collaborative writing and to emphasize the understanding of appropriate terminology. A total of 37 glossary terms was produced. The collaborative construction of glossaries in massive open online courses is not so common as it is in traditional educational implementations due to technological limitations of MOOC providers to support massive student contributions and a deeper learning analysis.

IV. LESSONS LEARNED

This section details some lessons learned from the Mapping of MOOCs presented in Section 2, and the design and delivery of the Agile Software Development MOOC described previously. They are related to the desired characteristics for MOOC Learning Maps as summarized in Table 4. Some technological and pedagogical issues of designing MOOCs to support SEE are also described.

A. Competency-based design

When developing MOOCs, instructors must review and consider essential competencies and skills for Software Engineers and students. Skill competencies, personality, and attitude factors will help them to best define active learning strategies, learning goals to be achieved by students, designing of learning maps, and development of contextualized videos, among others. Videos must guide and support students in carrying out activities that promote the development of those skills, competencies, and personal factors. For example, considering current results, Project-Based Learning is appropriate to support teaching in Computing Essentials; Software Modelling and Analysis; and Requirements Analysis and Specification. Case-based Learning could be used to teach Software Verification and Validation; Software Quality; and Security. Role-Playing Courses can be useful to work Software Process and Project Management.

B. User-centered learning

One of the essential elements of innovative design for MOOCs is considering activities that promote user empowerment and engagement. Instructors may use activities that activate student's past experiences with software development, use of methods, techniques, and their current knowledge about course's content. The MOOC syllabus can be identified through a survey of potential users and professional software engineers. As identified in this study, students enjoyed taking our MOOC at their own pace and this is another point to be considered by instructors.

C. Self-Regulation Learning

Many strategies can be used to support students in developing self-regulation skills, as defined in [21]. However, as identified in Section 2, most of the analyzed Software Engineering-Based MOOCs still do not consider such type of activity. In our MOOC, we used a "pre-start time" to encourage self-regulated attitudes towards learning. Nevertheless, students had difficulty to manage time, define learning goals, and control the own learning process. Therefore, this topic deserves special attention by instructors and demands further investigation. MOOC platforms need to consider new features to address this question.

D. Social Networking and Collaborative Learning

Communication and collaborative skills are essential in any area. In the IT sector, they are significant contributors in building effective interpersonal, and harmonious working relationships [11]. Our course attempts to motivate those skills through remote peer programming activities that simulate real contexts of agile development and extreme programming. However, technological and cultural gaps need to be considered. According to [3], more than 70% of self-paced distance learners seem to be open to peer collaboration but prefer working independently.

E. Deep Learning, knowledge creation and sharing

Activities that motivate top-level thinking skills development [13], such as drawing, constructing, scheduling, criticizing, experimenting, moderating, comparing and organizing are extremally important to promote deep learning and to create and share knowledge. Software engineering courses can benefit from this once they are related to project development, use of methods and techniques to build something and to test software products.

F. Assessment and Understanding

Instructors should prioritize formative rather than summative assessments. Self-paced activities based on well-defined rubrics, which stimulate auto-reflection should also be considered. Assessments must be authentic, based on real contexts and scenarios. For this, MOOC-instructors should strengthen their relationship with professionals Software Engineers. Such professionals could record contextualized video-interviews, attend webinars, help to define MOOC syllabus, write cases and scenarios for Case-based Learning, among other tasks. Also, the development of collaborative activities, such as glossaries in massive and virtual scale can be used to support other ways of formal and informal opportunities for student's assessment. However, MOOC platforms still need to provide better resources to collect and analyze massive data, such as who wrote the glossary terms, when, and how to evaluate the quality of the entry through peer ratings and comments, among others.

G. Accommodate differences

A Software Engineering-based MOOC should be open for students with different levels of background (i.e., beginner, specialist), knowledge in Programming Languages (i.e., C, Java), and needs (i.e., academy, labor market, research). The face-to-face course that inspired the MOOC presented in this study requires knowledge in very specific technologies: Ruby on Rails, Git, and Docker. In the corresponding MOOC, some adjustments were needed. The unique requirement was to know how to program, do not matter the language.

H. Feedback

Past MOOCs delivered in our university followed a self-paced strategy with no instructor support. Students did not receive e-mail contact, suggestion or help. They just had a forum and the support of course mates. Social presence was low and this brought negative results for those courses. On the other hand, the MOOC presented in this study provided feedback from instructors, peers, and automatically (i.e. from the platform). E-mails were often sent to students to welcome and to give them detailed explanation about complex activities. Instructors and tutors also maintained a social presence in the forum and Facebook group.

V. Conclusions And Future Work

This paper has presented a big picture of MOOCs in the context of Software Engineering Education and analyzed them from the perspective of the design for learning. Instructors and learning designers can use empirical evidence and guidelines presented in this work to improve course design and to share good practice in different software engineering topics.

Our findings show that several technological and pedagogical challenges still need further investigation to enhance appropriate learning in SEE. Technological issues include the development of tools integrated into MOOCs and learning analytics to support motivation, user experience, and more active learning experiences on specific topics (e.g. Project Management, Agile Methods, and Requirements Engineering). In turn, the most significant and needed improvement to the pedagogical discussion is the re-thinking of the virtual moment in the MOOC platform to optimize it using active learning activities.

On the other hand, this study has some limitations that should be considered when interpreting the related results. Firstly, the MOOC was delivered in Portuguese and without subtitles for foreign languages. It impairs the opening characteristic that should be present in MOOCs and also affects the global analysis of the situation. Secondly, of course, the mapping given in Section 2 did not consider all the available MOOC platforms around the world. Such limitations, however, do not invalidate the relevant results observed. They will be considered in our next MOOCs.

Our work in this area is ongoing. Two MOOCs in the Software Engineering and Programming context are under development: (i) Web Development using Agile Practices and (ii) Software Testing. Also, we plan to continue investigating new methods and strategies to support lifelong learning in Software Engineering through open and online courses.

Acknowledgements

The authors are grateful to the Brazilian funding agencies (CNPq, Capes, Fapemig, Fapesp) and the IFSULDEMINAS for their financial support. We also thank the contributions of Matheus Haddad and the *Ateliê de Software*'s team for helping us during the development and delivering of the Agile Software Development MOOC.

REFERENCES

[1] D. F. Rico and H. H. Sayani, "Use of agile methods in software engineering education," In: Agile IEEE Conference, pp. 174-179, 2009.

[2] C. D. O. Melo, V. Santos, E. Katayama, H. Corbucci, R. Prikladnicki, A. Goldman, and F. Kon, "The evolution of agile software development in Brazil," Journal of the Brazilian Computer Society, pp. 523-552, 2013.

[3] The New York Times, "The Big Three, at a Glance," November, 2012. Available at http://www.nytimes.com/2012/11/04/education/edlife/the-big-three-mooc-providers.html

[4] Skilledup for Learners, "The Best MOOC Provider: A Review of Coursera, Udacity and EdX," June, 2014. Available at http://www.skilledup.com/articles/the-best-mooc-provider-a-review-of-coursera-udacity-and-edx

[5] Association for Computing Machinery, "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering," February, 2015. Available at https://www.acm.org/education/se2014.pdf.

[6] A. G. O. Fassbinder, M. Fassbinder, E. F. Barbosa, and G. Magoulas, "Towards a MOOC Design Model based on Flipped Learning and Patterns: A Case on Introductory Courses," In: XXI Congreso Internacional de Informática Educativa (TISE), pp. 130-141, 2016.

[7] 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 Distributed Learning, v. 14, n. 3, pP. 202-227, 2013.

[8] D. Comier, "The CCK08 MOOC Connectivism Courser, October, 2008, Available at http://davecormier.com/edblog/2008/10/02/the-cck08-mooc-connectivism-course-14-way/

[9] E. Figueiredo, "On the evaluation of an open software engineering course," In: Frontiers in Education Conference (FIE), 2014.

[10] A. G. O. Fassbinder and E. F. Barbosa, "Design and Validation of an Instrument to Evaluate Learning Maps in the MOOC Context", Under review.

[11] C. D. Manawadu, M. G. M. Johar, and S. S. N. Pereira, "Essential Technical Competencies for Software Engineers: Perspectives from Sri Lankan Undergraduates," International Journal of Computer Applications, v. 113, n. 17, 2015.

[12] L. Toetenel and B. Rienties, "Analysing 157 learning designs using learning analytic approaches as a means to evaluate the impact of pedagogical decision making," British Journal of Educational Technology, 2015.

[13] D. R. Krathwohl, "A revision of Bloom's taxonomy: An overview. Theory into practice," v. 41, pp. 212-218, 2002.

[14] Common Wealth of Learning, "Guidelines for open educational resources (OER) in higher education", 2011. Available at unesdoc.unesco.org/images/0021/002136/213605e.pdf

[15] L. Stuchlikova and A. Kosa, "Massive open online courses-Challenges and solutions in engineering education," In: 11th International Conference on Emerging eLearning Technologies and Applications (ICETA), pp. 359-364, 2013.

[16] F. J. García-Penalvo and R. Colomo Palacios, "Innovative teaching methods in Engineering," 2015. Available in https://gredos.usal.es/jspui/bitstream/10366/126531/1/GRIAL_Innovative_teaching_methods_Engineering.pdf

[17] T. R. Liyanagunawardena, P. Parslow, and S. Williams, "Dropout: MOOC participants' perspective," EMOOCs, pp. 95-100, 2014.

[18] A. G. O. Fassbinder, M. Fassbinder, and E. F. Barbosa, "From flipped classroom theory to the personalized design of learning experiences in MOOCs," In: Frontiers in Education Conference (FIE), 2015.

[19] P. Goodyear, "Educational design and networked learning: Patterns, pattern languages, and design practice," Australasian Journal of Educational Technology 21 (1) , pp. 82-101, 2005.

[20] S. Warburton and Y. Mor, "A set of patterns for the structured design of MOOCs," Open Learning: The Journal of Open, Distance and e-Learning, v. 30, n. 3, pp. 206-220, 2015.

[21] C. E. Weinstein, J. Husman, and D. R. Dierking, "Self-regulation interventions with a focus on learning strategies," 2000.