DOI: 10.1109/educon.2016.7474695

From Software Engineering to Courseware Engineering

Carlos Delgado Kloos, Mª Blanca Ibáñez, Carlos Alario-Hoyos,
Pedro J. Muñoz-Merino, Iria Estévez Ayres, Carmen Fernández Panadero, Julio Villena
Dep. Telematic Engineering, Universidad Carlos III de Madrid, 28911 Leganés (Madrid/Spain)
{cdk, mbibanez, calario, pedmume, ayres, mcfp, jvillena}@it.uc3m.es

Abstract—**The appearance of MOOCs has boosted the use of educational technology in all possible contexts. As a consequence, many teachers are creating a lot of educational content (*courseware*) to be offered in MOOCs. Although some best practices exist, it is true that most of the content is being developed without much thought about adequacy, reusability, maintainability, composability, etc. The main thesis at this paper is that we are facing a "courseware crisis" in the same way as there was a "software crisis" 50 years ago, and that the way out is to identify good engineering discipline to aid in the development of courseware. We need *Courseware Engineering* in the same way as at those times we needed *Software Engineering*. Therefore, the challenge is now to define and develop fundamentals, tools, and methods of Courseware Engineering, as an analogy to the fundamentals, tools, and methods that were developed in Software Engineering.**

Keywords- MOOCs, SPOCs, courseware crisis, courseware engineering

I. COURSEWARE CRISIS?

In the 1960s, software was created without much discipline. The software created did often not satisfy the specifications. It had errors and was slow; it was of bad quality. Many projects didn't finish on time and on budget. Moreover, the development process often led to software that was difficult to maintain and evolve. This led to coining the term "software crisis". Software development had to be taken more seriously. A conference was called out in Garmisch-Partenkirchen (Germany) in October 1968, organized by F.L. Bauer from Technische Universität München, whose topic was the then recently coined term "software engineering". Bauer said then: "The time has come to switch from home-made software to manufactured software, from tinkering to engineering –twelve years after a similar transition has taken place in the computer hardware field. […] It is high time to take this step. The use of computers has reached the stage that software production […] has become a bottleneck for further expansion" [1]. Software Engineering is now an established discipline with well-defined fundamentals, numerous support tools, and rich methods.

Fast forward to today. Some universities have embarked in developing educational content (*courseware*) for MOOCs (Massive Open Online Courses) and SPOCs (Small Private Online Courses), their on-campus counterparts. Although some best practices exist, we are still at the beginning of perfectly understanding the power and the possibilities of using video-based instruction, rich interactions, and social components to its full potential. As we see, the MOOCs that are being offered today, even the best and most successful ones, they are still designed in a very much ad hoc way. How much MOOC content has been designed for maintainability and evolution? How well the diversity of learners worldwide has been taken into account (apart from the multilingual transcripts)? What fundamentals have been used to decide which means to use to teach concrete concepts? Is content related to assessment in an ad hoc way?

We believe that it is fair to say that now that the technological affordances to educate have exploded, we find ourselves in a "courseware crisis" in a similar way that we had a "software crisis" 50 years ago. We are in need not only of a good instructional design underpinning; we also need to take care of the whole life cycle of educational material. We need to think about courseware evolution. We need to have instruments to make content more robust. We need to understand how to make educational material adequate for diverse learners. We need a strong "courseware engineering" discipline, in the same way as "software engineering" disciplines are well established.

Of the several aspects that can be analyzed in courseware engineering, this paper focuses primarily on three of them: fundamentals (section II), tools (section III) and methods (section IV). The next sections will therefore discuss the existing challenges related to the definition and development of fundamentals, tools and methods for courseware engineering, making an analogy with those challenges that were already tackled and that currently present mature solutions in software engineering.

II. COURSEWARE ENGINEERING: FUNDAMENTALS

Let's consider three important properties of computer programs: *correctness*, *efficiency*, and *usability*. We want programs to do the right thing (*correctness*), with as few resources as possible (*efficiency*), and allowing end-users to easily interact with them (*usability*).

These properties gave rise to important bodies of knowledge (fundamentals), such as semantics, complexity theory and human-computer interaction (HCI). We want to know the mathematical function a program corresponds to. Therefore, several semantic theories were developed (denotational semantics, operational semantics, axiomatic semantics, etc.). They allow mathematically checking a program against a formal specification. Since we know that this is often too complex, alternative procedures were identified to increment the confidence of the correctness of code. Testing methods were established for example to check that the program does what it has to do, or to check how it responds to possible unintended input. Another big field is complexity theory, which introduces mathematical models to quantify in an abstract way the amount of resources needed to solve computing problems. The time to solve the problem, or the storage needs are some of the metrics used for quantification purposes. HCI focuses on the definition of interfaces that facilitate the interactions between users and computers, with the aim to improve the usability of computer systems and programs.

What are the counterparts when we speak of courseware instead of software? We want courseware to be: *effective*, teaching the subject in the right way (as an analogy to correctness); *efficient*, having a low production and maintenance cost; and *usable*, being easy to use and learn. Thus, when creating courseware, we want learners to understand and assimilate the right thing in an easy way, no matter whether they are more visual, verbal, logical, etc., and we want to achieve this in an efficient way.

Courseware effectiveness implies the acquisition of knowledge and skills by students through interaction with educational resources (e.g., videos, exercises, and other activities). The effectiveness can be measured globally (for the whole course), or locally (for particular learning activities) [2]. Courseware effectiveness does not only relate to cognitive aspects, but also involves meta-cognitive skills, behaviors and emotions. In this direction, courseware can produce different behaviors or emotions that may affect positively or negatively the acquisition of knowledge.

Courseware efficiency is aimed at producing the same outcome (acquisition of knowledge and skills by learners) with the minimum number of resources. For example, two different learning materials can produce the same effect on students in terms of acquisition of critical thinking skills, but one of these materials might be easier to create and maintain, being this way more efficient than the other.

Courseware usability is aimed to facilitate the use of educational content by end-users. This can include for instance adding transcripts to videos to improve understanding and for accessibility reasons, enabling the download of low-resolution versions of videos for those learners with poor Internet connections, or adapting contents to be used from both laptops and mobile devices. But courseware usability can also be understood as facilitating learning, using multimodal resources (videos, texts, interactive activities, etc.) to reach students with different intelligence modalities [3].

A. *Mathematical representation of courseware*

Although there are many initiatives to model and standardize the educational process in general and courseware in particular, there have been very few attempts to formalize these processes mathematically. One of the few models that defines this mathematical formalization is EPM (Educational Practice Model) [4][5]. Objections to this model are due to the use of quantitative measures, such as the effectiveness and efficiency in a course, which conditioned by the number of students who interact with it. The rapid expansion of MOOCs in the last few years draws an excellent setting for new applications of EPM.

In the same way a program can be represented by a mathematical function, EPM allows representing courseware as an operator in a Hilbert space. The number of dimensions of the Hilbert space is defined by course objectives. The state of the student can be seen as a point in an n-dimensional space by considering her performance with respect to each objective. Due to the uncertainty of the student state, the evaluation process is seen as an estimation of its real position. The effectiveness of the course will be represented by the operator's ability to translate a number of points (representing students) to the desired region of the Hilbert space. This region will be delimited by the range of expected values of performance in each course objective. Course efficiency is related to the speed at which this transformation occurs. This speed can be calculated taking into account the time of application of the operator (course enactment) or can include also the time spent on its construction (course design). All the formalisms use the abstraction process followed by Physics to unify the representation of the fundamental interactions using Hilbert spaces.

A computer program once written produces the same result on different computers, even with different architectures. However, the same courseware could have different effects on different people. The parallelism with Physics is useful also in this case because it allows modeling this complexity, introducing the concept of mass (resistance to being accelerated). In our case, students, depending on their characteristics, can present different inertia to change their state of knowledge, i.e. to move from a point in the Hilbert space (initial state of knowledge) to other (final state of knowledge) by the action of an operator (courseware or any other educational intervention).

III. Courseware Engineering: Tools

Computer-aided software engineering (CASE) tools underpin the creation and management of computer programs, supporting their *design*, *development*, *testing*, as well as the *management* of the whole software life cycle. CASE tools are useful for applying development processes and frameworks in a methodical way. They allow working collaboratively (as most of the software is currently done in teams). And they support both classical (e.g., waterfall model) and agile methodologies.

CASE tools aimed at facilitating the *design* of software allow for instance the creation of drafts and detailed diagrams on how the software must behave and is structured. Examples of design tools are ArgoUML [6] or BOUML [7], which allow creating both behavioral UML diagrams (e.g., use case and sequence diagrams) and structural UML diagrams (e.g., class and component diagrams). Balsamiq [8] is another example of design tool for creating agile mockups of graphical user interfaces for desktop, web and mobile applications. These diagrams and mockups need to be later translated to different programming languages; fortunately it is possible to generate code automatically (or semi-automatically) for instance from the diagrams and mockups created through the aforementioned CASE tools, reducing this way the *development* time and workload. Once the software is developed, we need to *test* that it is correct, efficient and bug-free. For example, tools such as Java Pathfinder [9] or FindBugs [10] will help us detect bugs and deadlocks in programs written in the Java language. Finally, it would be also convenient to use project *management* tools to support communication, collaboration and version control during the creation of the computer program. Basecamp [11] or Trello [12] are project management tools with features for message exchange, file sharing, milestone management or time tracing. GIT [13] is an example of version control system that acts as a shared repository and solves the problems of concurrency and version control when developing software in teams.

The same analogy can be applied to courseware, as there is also a need for *designing*, *developing* and *testing* courseware in an efficient way, and we face the same problems of *managing* the courseware life cycle in MOOCs and SPOCs. It is therefore necessary to find appropriate computer-aided courseware engineering (CACE) tools to support faculty and staff in the creation and management of courseware.

When *designing* courseware it is important to take into consideration that most MOOCs and SPOCs are done by groups of teachers and therefore we need to facilitate collaboration and co-creation in the design of the courseware. The MOOC Canvas [14] is a tool that allows sharing an overview of the design of the MOOC or SPOC considering the main available resources and the key high level design decisions, including the overall structure of the course and the assessment system, among others. Of course, we can also design and refine the structure of the course directly on the platform in which it is going to be deployed, following a more agile approach known as "bricolage". Studio [15], the authoring system of edX and Open edX [16], allows a bricolage-like design of MOOCs and SPOCs, assembling low-level components (e.g., videos, exercises, etc.) in a four-level courseware structure. The design of courseware also includes the generation of the particular slides and scripts that will be used in the creation of videos and exercises. Collaborative text editors and presentations tools such as Google Documents and Slides may be useful for this purpose. From a pedagogical point of view, the course design also involves the orchestration of the educational process. Orchestration entails deciding at what point it is better to introduce different types of activities (memorization, application, creation), mechanisms of collaboration, or gamification elements (points, badges, leaderboards, competition, etc.) with the aim to enrich the educational process. For example, the PhyMEL methodology (Physical Mental and Emotional Learning) [17] uses the universal narrative pattern of the Hero Journey described by Campbell [18] and Vogler [19] to help students in each stage of their particular journey. Nevertheless, there are still many research challenges in the design of courseware, such as the development of context-driven editors for creating scripts and slides; given a context, e.g., a course about programming in Java, a context-driven editor should be able to facilitate writing intertwined pieces of Java code and plain-text explanations in the script or slides.

The *development* of courseware, especially the one that includes audiovisual content, can be a burdensome task. There is a need for low-cost and efficient video production tools that allow creating ready-to-go videos from raw footage in a very short time after the teacher records a lecture in the studio. The Polimedia tool [20], developed by Universitat Politècnica de València (UPV), is a successful technology for the efficient creation of multimedia contents that does not require teachers to have much technical or audiovisual background. The production of transcripts in different languages for videos typically demands also a high workload, but these are a must in MOOCs due to the wide audience and for accessibility reasons; Translectures [21], also by UPV, is a tool for the automatic transcription and translation of educational videos that highly reduces the time to generate transcripts. Once videos and exercises are developed, they need to be deployed in the platform or platforms where they are going to be consumed by learners; GE-L+ [22] is a Content Management System by Universidad Carlos III de Madrid (UC3M) that allows structuring courseware and afterwards automatically deploy it in edX and in Open edX local instances. There are however many research challenges in the development of courseware, such as the generation of high-quality low-cost videos that include animations and built-in interactive activities by teachers without much technical background.

The *testing* of courseware affects error detection prior to course delivery, but also the correction of errors during and after course delivery and the improvement of courseware for subsequent editions of the MOOC or SPOC. Nowadays there are powerful learning analytics tools that e.g., allow detecting questions that are not correct or that may be confusing, and videos that students need to watch several times, as the concepts are not clearly enough explained on them. Insights by edX or ANALYSE and ALAS-KA by UC3M [23][24] are example tools that provide detailed visualizations aimed at helping teachers improve their MOOCs and SPOCs in edX and Open edX. For example, questions that are not solved correctly by any of the students, or that have a low success rate, can be detected and further analyzed. Parts of videos that students watch several times may be analyzed with the aim to infer if these parts should be improved; or parts of videos that students do not watch may be analyzed to infer if there is some reason that makes them demotivating for students. In addition, the testing of courseware may involve general indicators, such as learning gains using a pre-test and post-test, or surveys to measure students' satisfaction and motivation with the course in general, or with specific learning resources and activities in particular. There are still research challenges in the testing of courseware, such as tools for the generation of overall reports for teachers indicating the high level topics that are not sufficiently covered in the course and the concepts that, although essential to follow the course from scratch, are missing and need to be introduced somewhere in the course. In addition, although there are tools to analyze students' behaviors, emotions and meta-cognitive skills in general, there is a need of tools that provide a detailed analysis of the differences of educational resources regarding what they produce in terms of behaviors, emotions or meta-cognitive skills.

The *managemen*t of the courseware life cycle shares with the management of the software life cycle the same problems of supporting communication between the teaching staff, version control of educational resources, milestone management, etc. Project management tools, such as Basecamp or Trello may be useful to follow the development process of the courseware in an efficient way, allowing the communication between teachers, and the follow-up by a project manager. Nevertheless, there are still open challenges regarding version control of audiovisual resources and concurrency (e.g., editing several parts of the same course at the same time in authoring tools such as Studio, in edX).

All in all, CASE and CACE tools share common purposes: helping professionals in the design, development, testing, and management of products (either software or courseware). Ultimately CASE and CACE tools are intended to foster efficiency not only in the final product, but also in the creation and management processes.

IV. COURSEWARE ENGINEERING: METHODS

There are many software development paradigms and methods. Let's simplify them by just concentrating in two families: *waterfall* and *agile*.

The waterfall model for software development proposes a sequential design process, in which a phase is not initiated until the previous is not finished (see Figure 1). So, for instance the design phase does not start until the requirements phase is completed. Reverting back to an earlier phase is always very costly. If when verifying the designed software, you decide to change some requirements, this will cost you a lot in terms of effort.

If you look at MOOC development methods around the world, they normally follow a waterfall model, as it is the case, for instance, in The University of Queensland [25] or in The University of British Columbia [26], among many others. In the waterfall model, the verification of courseware is usually done at a late stage, usually when there is little time left before releasing the MOOC or SPOC. In addition, content is normally not produced thinking of facilitating corrections or modifications (e.g., videos are costly to modify once they are produced). Therefore, if errors are identified at this very late stage, one has to repeat many steps, and even in some cases to put off the start of the course.

To solve some of the problems of the waterfall model in software development, agile software development methods have been introduced. Agile software development promotes that requirements and solutions evolve together through collaboration; and design and testing occur in rapidly alternating steps (see Figure 2). Agile software development dates back to February 2001, when 17 people from various software companies acknowledging human fallibilities in software development and declared the 4 values and 12 principles that came to be known as "Agile." The four values were called The Agile Manifesto [27]: 1) individuals and

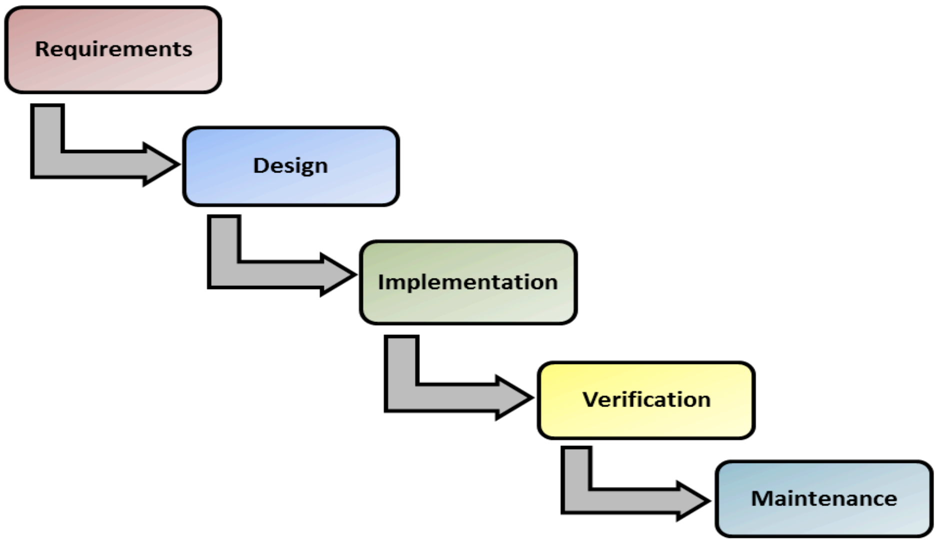

Figure 1: Waterfall model for the creation of software, which has also been adopted for the creation of courseware.

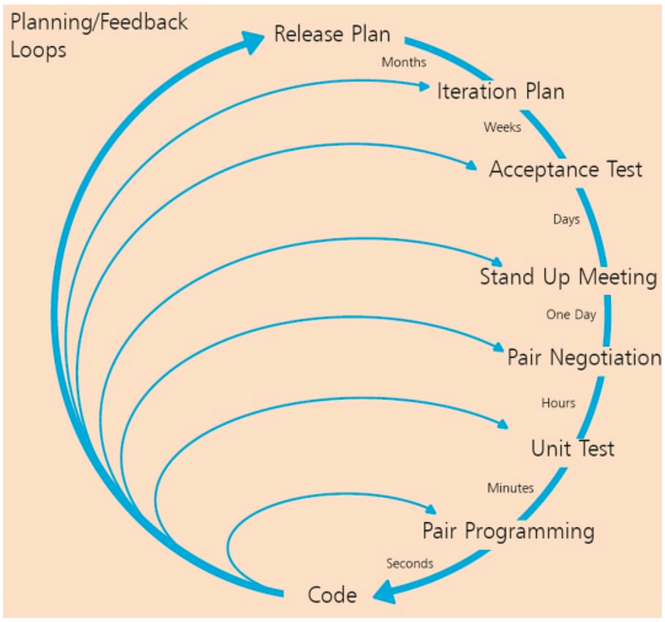

Figure 2: Agile model for the creation of software (Extreme Programming), which can also be adopted for the creation of courseware. Figure taken from [28].

interactions over processes and tools; 2) working software over comprehensive documentation; 3) customer collaboration over contract negotiation; and 4) responding to change over following a plan.

In addition to the philosophical values of the manifesto, 12 principles underlying it were articulated: 1) customer satisfaction by early and continuous delivery of useful software; 2) welcome changing requirements, even in late development; 3) working software is delivered frequently (weeks rather than months); 4) close, daily cooperation between business people and developers; 5) projects are built around motivated individuals, who should be trusted; 6) face-to-face conversation is the best form of communication (co-location); 7) working software is the principal measure of progress; 8) sustainable development, able to maintain a constant pace; 9) continuous attention to technical excellence and good design; 10) simplicity —the art of maximizing the amount of work not done— is essential; 11) self-organizing teams; and 12) regular adaptation to changing circumstance.

How might this manifesto, written for the context of software development, be rewritten for the context of a courseware development? Is it possible to draw an analogy, or at least a preliminary set of good practices? In this regard, some reflections might be proposed:

R1. What should be produced as output?

R2. What would be measured, assuming that it is possible or feasible?

R3. Who are the "customers"? And who are the stakeholders?

A MOOC can be seen as a web-class environment aimed at a large-scale global participation and open access via the Web. MOOCs tend to couple courseware in multiple formats: text, video, assignments, exams, etc. The learning content is accessible through a cloud-based computing system around which communities of learners and content publishers assemble and interact. MOOCs are generally offered as standalone courses but can be integrated into an organized curriculum as SPOCs. Quality of MOOCs should be measured mainly by the quality of its courseware in terms of educational value. In this regard, Conole [29] states that good learning: encourages reflection, enables dialogue, fosters collaboration, applies theory learnt to practice, creates a community of peers, enables creativity, and motivates learners.

MOOCs' stakeholders include: content publishers and curators, professors, and students. Among the most important MOOC publishers and curators are edX [16], Coursera [30], and Udacity [31]. They perceive that MOOCs can help universities to reduce operating costs, students by making education more affordable, better, and accessible and future employers, by providing data on students' performance and credentials in designated areas of study [32]. Faculty can benefit by folding MOOC courseware into their regular courses, and professors who produce and deliver MOOCs might receive acclaim, visibility and eventually ways to monetize their fame. Students are both consumers of the MOOC content and potential labor for peer assessment and content reviewers.

Like a software project, courseware can be difficult to develop, the process can be inefficient, as can the learning that they intended to achieve, and they can be error prone in failing to achieve the desired outcomes. Courseware developers are face to the challenge of producing engaging materials aimed at improving students' knowledge and skills. If we focus in the student as customer, the *first principle* could be rewritten as:

"*Learning satisfaction through early and continuous delivery of valuable learning experiences*"

MOOCs, as any course, have difficulties to measure and transmit students their actual progress in their learning process. Fortunately, learning analytics is making progresses to contribute in this regard, enriching this way learners' experience [2].

The *second* principle might be stated as:

"*Welcome changing learning needs, even close to the end of the course*"

This principle is hardly acceptable in a formal educational process and even in MOOC courses where an initial planning is done according to the learning outcomes desired. However, MOOCs have the advantage over traditional courses of having the freedom of being delivered using smaller chunks of information. This has an obvious similarity with agile software development techniques where there are a concern to build functional limited but still useful artifacts in short periods of time. Therefore, MOOCs courseware development might also fit the *third* agile principle. The *fourth* principle is difficult to satisfy in MOOCs but students' feedback can be very helpful to adjust parts of the courseware according to new customer requirements. The atomicity of units of information and assessment is key to guarantee the agile principles (7) - (9) and (12). Finally, the rest of principles have to do with the courseware developing team and they are not the object of this study.

The debate on the use of agile methods in the field of education is open, and does not only affect the production of courseware. The NMC Horizon Report already highlights the need for incorporating agile approaches to change in education in the 2014 Higher Education Edition [33], and suggests that Higher Education Institutions need to adopt startup models for a more efficient implementation of new practices and pedagogies. We believe that the methods used so far for the production of courseware are very much waterfall oriented, and that faculty should adopt agile methods as a way to accelerate and improve MOOC production.

V. CONCLUSIONS

The parallel between courseware and software gives useful hints in what is needed to develop courseware to its full potential in the current rapidly changing educational landscape where MOOCs and SPOCs threaten to shake the foundations of traditional educational systems. Fundamentals, tools and methods from the scope of software engineering may have a correspondence in the scope of courseware engineering. But there are many other similarities that can be found. For example, MOOCs are normally announced a long time before they start. This allows the registrations to grow. But often the course is not finished at the time of announcement and therefore there are very tight development timelines; as it happens in most software projects were work is done under time pressure.

Nevertheless, the parallel between software engineering and courseware engineering should not be taken too far. A computer program once written should run well on different computers, even with different architectures. However, the same courseware could have different effects on different people. Therefore in education, the context acquires a higher importance, because it cannot be factored out so easily. There is even discussion whether learning styles really exist. So, the field is much more complex here. However to have more rigor and assistance for the creation of courseware should be of great help.

ACKNOWLEDGMENT

The eMadrid Excellence Network is being funded by the Madrid Regional Government (Comunidad de Madrid) with grant No. S2013/ICE-2715. This work also received partial support from the Spanish Ministry of Economy and Competitiveness Project RESET (TIN2014-53199-C3-1-R) and from the European Erasmus+ projects MOOC Maker (561533-EPP-1-2015-1-ES-EPPKA2-CBHE-JP) and SHEILA (562080-EPP-1-2015-BE-EPPKA3-PI-FORWARD).

REFERENCES

[1] D. MacKenzie, *Mechanizing Proof: Computing, Risk and Trust*, Cambridge, MA: MIT Press, 2001.

[2] P. J. Muñoz-Merino, J. A. Ruipérez-Valiente, C. Alario-Hoyos, M. Pérez-Sanagustín, C. Delgado Kloos, "Precise Effectiveness Strategy for analyzing the effectiveness of students with educational resources and activities in MOOCs," *Computers in Human Behavior*, vol. 47, pp. 108-118, 2015.

[3] H. Gardner, *Multiple Intelligences: New Horizons in Theory and Practice*, Basic Books, 2006.

[4] C. Fernández-Panadero, A. Pardo, J. Fernández-Panadero, A. Marín López, "A mathematical model for reusing student learning skills across didactical units," *Proc. 2002 Frontiers in Education, FIE 2002*, pp. F1A-1-F1A-6, 2002.

[5] C. Fernández-Panadero, "EPM A Mathematical Model for Characterization and Diagnosis of Educational Processes," Ph.D. dissertation. Universidad Carlos III de Madrid, 2004. Note: In Spanish.

[6] ArgoUML, argouml.tigris.org

[7] BOUML, bouml.fr

[8] Balsamiq, balsamiq.com

[9] Java PathFinder, javapathfinder.sourceforge.net

[10] FindBugs, findbugs.sourceforge.net

[11] Basecamp, basecamp.com

[12] Trello, trello.com

[13] Git, git-scm.com

[14] C. Alario-Hoyos, M. Pérez-Sanagustín, D. Cormier, C. Delgado-Kloos, "Proposal for a conceptual framework for educators to describe and design MOOCs," *Journal of Universal Computer Science*, vol. 20, no. 1, pp. 6-23, 2014.

[15] edX Studio, studio.edx.org

[16] edX, edx.org

[17] C. Fernández-Panadero, C. Delgado Kloos, "PhyMEL. A Framework to Integrate Physical, Mental and Emotional Learning in Meaningfull Experiences and Multidimensional Reports," *Proc. 3rd European Immersive Education Summit*, pp. 203-208, 2013.

[18] J. Campbell, *The hero with a thousand faces*. Pantheon Books, 3rd ed., 2008.

[19] C. Vogler, *The Writer's journey*. 3rd Revised edition ed. Michael Wiese Productions, 1998.

[20] C. Turró, M. Ferrando, J. Busquets, A. Cañero, "Polimedia: a system for successful video e-learning," *Proc. Eunis 2009 International Conference*, 2009.

[21] M. A. del Agua, A. Giménez, N. Serrano, J. A. Ferrer, J. Civera, A. Sanchis, A. Juan, "The translectures-UPV toolkit," *Advances in Speech and Language Technologies for Iberian Languages*, pp. 269-278, 2014.

[22] C. Delgado-Kloos, P. J. Muñoz-Merino, M. Muñoz-Organero, C. Alario-Hoyos, M. Pérez-Sanagustín, H. A. Parada G., J. A. Ruiperez, J. L. Sanz, "Experiences of Running MOOCs and SPOCs at UC3M," *Proc. 2014 IEEE Global Engineering Education Conference, EDUCON 2014*, pp. 884-891, 2014.

[23] P. J. Muñoz-Merino, J. A. Ruipérez-Valiente, C. Delgado Kloos, "ANALYSE: A learning analytics extension for Open edX," *Open edX Conference*, 2015.

[24] J. A. Ruipérez-Valiente, P. J. Muñoz-Merino, D. Leony, C. Delgado Kloos, ALAS-KA: "A learning analytics extension for better understanding the learning process in the Khan Academy platform," *Computers in Human Behavior*, vol. 47, pp. 139-148, 2015.

[25] UQx Design Methodology for Course Development, *Technical Report*, The University of Queensland, Australia, 2015. http://uqx.uq.edu.au/filething/get/533/UQx%20Methodology%2020%20May%202015.pdf

[26] Design Quality Online Course, Technical Report, The University of British Columbia, Canada, 2015. http://wiki.ubc.ca/Design_Quality_OnlineCourse

[27] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, et al. Manifesto for Agile Software Development, agilemanifesto.org

[28] J. Carroll, *Agile Project Management in easy steps*, In Easy Steps Limited, Warwickshire, UK, 2012.

[29] G. Conole, "MOOCs as disruptive technologies: strategies for enhancing the learner experience and quality of MOOCs," *RED, Revista de Educación a Distancia*, vol. 39 pp. 1-17, 2013.

[30] Coursera, coursera.com

[31] Udacity, udacity.com

[32] B. Dasarathy, K. Sullivan, D. C. Schmidt, D. Fisher, A. Porter, "The past, present, and future of MOOCs and their relevance to software engineering," *Proc. Future of Software Engineering*, ACM, pp. 212-224, 2014.

[33] NMC Horizon Report: 2014 Higher Education Edition, Technical Report, New Media Consortium and EDUCAUSE Learning Initiative, 2014. nmc.org/pdf/2014-nmc-horizon-report-he-EN.pdf