Applying Software Engineering to Teach Online Project-Based Software Engineering Courses

Duc Minh Le

Department of Information Technology, Swinburne Vietnam, FPT University

Abstract

Software engineering (SE) courses are facing increased challenges from emerging online learning platforms. Teaching project-based SE (PBSE), however, remains complex because SE development itself is complex, broad and requires constant practice to master. In this work, we propose to apply SE to teach online PBSE courses. We present the instructional design model of PBSE course, which revolves around an iterative software development model that uses a software framework to help students practise software development. We discuss our experience in applying the course model to a recent PBSE course. The course was delivered in a hybrid mode in order to adapt to the frequent Covid-19 lockdowns in Hanoi, Vietnam. The course uses an integrated LMS consisting of the Google and GitHub classroom platforms. The former helps manage the student class and teaching materials, the latter helps students learn to collaboratively work on and manage a software project online.

Keywords

educational technologies and tools, software engineering method, online teaching method

1. Introduction

Software engineering (SE) is challenging because it involves not only the state-of-the-art engineering principles but knowledge about human and organisational behaviours. Software development therefore requires a wide range of technical and social skillsets. The main challenges with SE education are to provide learners with not only the knowledge but an authentic context in which they can practice what they learn. Access to the latter has been particularly difficult under the recent Covid-19 crisis, because most of the human and organisational activities have been performed online.

Inspite of these, it is interesting to observe from recent studies [1, 2] how SE education can relatively easily be adapted to online teaching. It can be argued that the "non-physical" nature of SE [3] helps make it more suitable for online technology adoption than other engineering disciplines. This of course should not be taken for granted and SE course designers need to proactively research and apply suitable methods, techniques and tools to prepare their courses for online technology adoption.

Of particular relevance to our work is project-based SE (PBSE) course. A PBSE course uses a semesterwide project as a main teaching component. It is often used in advanced course levels to provide students with an opportunity to apply a software development process to solve real-world complex problems.

Teaching PBSE courses is hard because SE development itself is complex, broad and requires constant practice to master. In this paper, we discuss our proposal to apply SE to online PBSE courses. We define an instructional design model of PBSE course, which is supported by an iterative software development model. Software development practice is made more productive with the help of a software framework. We apply this course model in a recent PBSE course, which was taught in a hybrid mode (offline and online) to adapt to the frequent Covid-19 lockdowns in Hanoi, Vietnam. We reflect on some of the lessons learnt through this course, especially concerning the support relationship between software development cycle and the instructional design cycle of PBSE course and how this can be leveraged to improve the course design. The rest of the paper is structured as follows. Section 2 discusses the instructional design of online PBSE course. Section 3 presents a our experience report with a recent PBSE course. Section 4 concludes the paper.

Proceedings of 4th Software Engineering Education Workshop (SEED 2021) co-located with APSEC 2021, 06-Dec, 2021, Taipei, Taiwan
duclm20@fe.edu.vn (D. M. Le)
0000-0001-7654-1127 (D. M. Le)

2. Instructional Design

Raiser & Dempsey [4] define instructional design as "...a system of procedures for developing education and training materials in a consistent and reliable

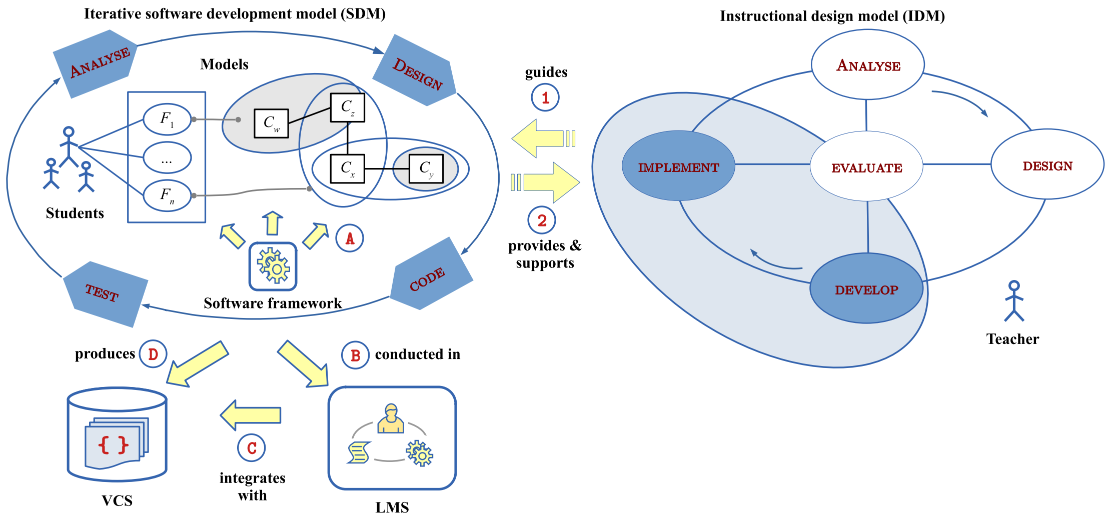

Figure 1: (RHS) An iterative instructional design model of online PBSE course (adapted from [4]), supported by (LHS) an iterative software development model.

fashion". In this section, we discuss the instructional design model of an online PBSE course.

2.1. Pedagogical Approach

The underlying pedagogical theory of our design approach is constructivism [4]. It is applied to PBSE as follows: (1-active) student's self-construction of a software that solves problems, (2-authentic) using authentic, real-world problem solving and (3-social) students collaboratively work on and manage software construction. In a PBSE course, students are constructivists who use various models to conceptualise, design and eventually write the software code in a target programming language. Models are particularly useful in earlier phases of the SE process, when the software concepts are still being shaped and potentially complex design rules need to be made precise in an incremental manner.

2.2. Iterative Instructional Design Model

In the popular instructional design model named ADDIE [4], the instructional design process is iterative and consists of 5 main phases: analyse, design, development, implement and evaluate. The analyse phase involves analysing the course requirements to determine the learning objectives. The design phase creates a detailed structure of the course, which includes the topics and their teaching sequence, and the assessment strategies. The develop phase consists in developing the materials for the defined topics and assessment items. The implementation phase concerns with the actual delivery of the course in a given teaching period. The evaluation phase involves carrying out the assessment items (both formative and summative) and, based on the results, reflect on whether the course has met its target objectives.

Through the lense of constructivism, we observed that in a PBSE the teacher's activities in two phases, namely develop and implement, involve defining only an overall framework and the core content. The remainder of the content are actually contributed by students, as they construct their software to solve problems. This observation led us to the instructional design model shown in Fig. 1. The figure shows the **instructional design submodel** (**IDM**) (on the RHS) and an iterative **software development submodel** (**SDM**) (on the LHS). SDM [3] consists of four main phases, which are performed iteratively to incrementally construct software prototypes. The arrows depict the relationships between the two submodels as well as the those between the components of SDM.

Arrow ① clearly shows how IDM guides SDM. The main actor of the IDM is teacher, while the main actors in SDM are students. Conversely, arrow ② shows how SDM provides student-constructed content for the two phases (develop and implement) of IDM. These phases are highlighted in the figure with a filled circle and are enclosed in an overlay filled oval. More than just the content, however, SDM provides SE techniques and tools that help perform the two phases more effectively. In particular, the figure highlights the use of **version control system** (**VCS**) and **learning management system** (**LMS**). In the develop phase, both the VCS and LMS are used by the teacher to author the contents needed for the course. In the implement phase, the software-related deliverables are distributed and shared on the VCS, while other course contents are delivered though the LMS.

Within the SDM, arrow (A) shows the role of a **software framework** in helping to carry out the SDM's phases more effectively. Software framework [3] helps make more informed and practical decisions about the structure of the software models that are created. Equipped with software generation capabilities, framework can help build software considerably faster. Arrow (B) represents the use of LMS to manage the delivery of the contents related to the SDM. Some LMSes provide plugins that support the execution of source code and provide feedbacks. More generally, LMS is used as a shared system for course delivery. Arrow (C) illustrates the integration between LMS and VCS. Arrow (D) depicts the various outputs that are produced by students when they perform activities in the SDM. Among the outputs include software source code as well as other artefacts (e.g. requirement definition and design documents).

As shown in Fig. 1, both IDM and SDM are iteratively performed but with their own cycles. An SDM's cycle typically consists in several development iterations, which are defined in the course's teaching plan and performed by the students to carry out their software projects. An IDM's cycle occurs per one course delivery, at the conclusion of which the course design is revised and improved for the next delivery.

2.3. Learning Objectives

Based on the IDM of PBSE course presented above, we define the following learning objectives for the course:

LO1. Study a software framework for developing software.

LO2. Investigate and define a real-world problem.

LO3. Apply an iterative software development method to plan, analyse, design, implement, and test a software solution for a defined problem.

LO4. Apply the software framework with the chosen method to develop software in a group project.

LO5. Apply a distributed VCS system to collaboratively work on and manage a software project.

The above objectives are defined in a manner that is not dependent on the technological platforms or programming languages used to implement the software. They are thus can be used as a guide to define objectives for PBSE courses at different levels. For lower-level courses, more specific objectives would be added to narrow the scope, focusing on a certain language or platform. For more advanced courses, the objectives can be used as such to provide an open scope for students to learn how to make technological decisions.

Objective *LO1* should be carried out first in the course in order to learn the software framework that will be used for software development in the project. The choice of framework depends on the software development state-of-the-art and on the suitability to the software development method. The remaining objectives aim to execute the SDM with the help of the software framework in group project.

2.4. Learning Activities

We categorise learning activities into two groups based on the scope: (I) inside class and (O) outside class:

A1. (O) Study the relevant coursebook chapters about the software framework and VCS

A2. (I) Collaboratively work on the chapter exercises about the framework and VCS

A3. (O) Collaboratively research a real-world problem to solve

A4. (O) Iteratively and collaboratively construct and manage software code and other deliverables using the distributed VCS

A5. (I) Present project progress in weekly seminars to obtain feedbacks from peers and teacher

A6. (O) Write technical documentation for the software. Reflect on what has been learnt.

Most of the learning activities are performed outside class in order to prepare for the class activities. For studying the framework, a *coursebook*[1] should

[1] coursebook is either formally written or a collection of relevant technical online documentations.

made available. There are typically no formal lectures in the course. Instead, in-class activities are designed to help student groups perform hands-on work on software development and for them to get feedbacks from peers and the teacher.

We observe that all of these activities can be conducted online via the LMS and a suitable VCS. Software development activities are carried out in an IDE (e.g. Eclipse, NetBeans, IntelliJ, *etc.*) and the deliverables are stored directly into the VCS.

2.5. Software Framework

It is a common practice to use a software framework in order to develop software faster. The productivity improvement is particularly higher in complex software, where similar design structures can be repeatedly applied to create different parts of the software. As shown in Fig. 1, we propose to use a software framework in order to help students practise and carry out software development more effectively.

It is essential that the software framework supports not only the design modelling decisions but also those that are made earlier in modelling the domain concepts. Frameworks that implement the domain-driven design (DDD) philosophy are particularly suited for this purpose. In DDD [5, 6], domain models are treated as the "heart" of software. DDD software frameworks (such as [7, 8, 9]), thus, have an important benefit of supporting domain modelling natively in a target programming language. We will discuss in Section 3 an example software framework, named jDomainApp [9], and how is used in a PBSE course delivery.

2.6. Integrating LMS with VCS for PBSE

It is essential for the LMS of a PBSE course to have some level of integration with the VCS. At the minimum, the integration should allow students to have the same identities when they login to use both systems.

For example, Google Classroom[2] and Meeting[3] form an LMS, while GitHub classroom[4] is a VCS. It was reported in [10] that GitHub classroom supports a feature that enables the import and synchronisation of the class roster with Google Classroom. This becomes a type of shared identity management service for both the LMS and VCS. We will show shortly in Section 3 how this feature is used in an online PBSE course.

[2] https://classroom.google.com
[3] https://meet.google.com
[4] https://classroom.github.com

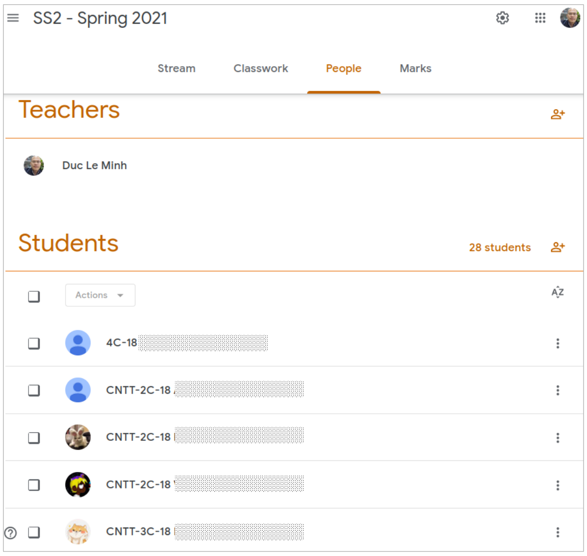

Figure 2: The Gclassroom's roster for class SS207. It is used to create and is synchronised with the GitHub classroom roster shown in Fig. 3.

3. An Advanced PBSE Course at Hanoi University

In this section, we briefly discuss an implementation of the IDM defined in the previous section for an advanced PBSE course.

3.1. PBSE Course

The PBSE course, named Special Subject 2 (SS2), is part of the Software Engineering major of the 3-year Bachelor of IT program of the Faculty of IT (Hanoi University, Vietnam). It is an advanced software engineering course that was taught in the Spring-2021 semester (the second semester of the second year). The student class was named SS207 and had 26 students. All students had previously completed an introduction software engineering course. The course adopted Java as the programming language for software development.

In previous semesters, this course was conducted in the standard offline (face-to-face) mode. However, during the Spring-2021 semester, Hanoi was undergoing two successive Covid-19 lock-downs and so we had to switch between offline and online teachings twice in this semester. Fortunately, our previous experience with other Covid-19 lockdowns had helped us in our preparation for the course. In the preparation, we looked for suitable online plat-

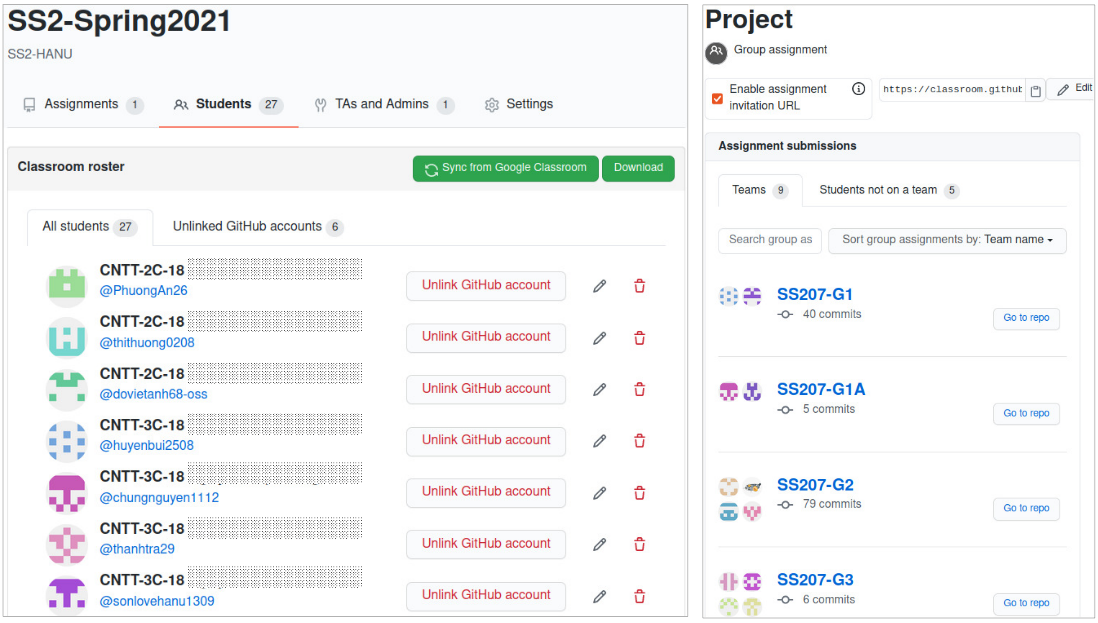

Figure 3: GitHub classroom setup: (1-LHS) the class roster (synched with Google classroom), (2-RHS) student project groups. The GitHub repository of each group (e.g. "`project−ss207−g6`") is named after the assignment name ("`Project`") and the group name ("`SS207−G6`").

forms (including LMS and VCS) and implemented the course on these platforms. In the period of the semester where the course was delivered face-to-face, the platforms also proved valuable for students and their groups to perform the outside class activities and be ready for the in-class discussion.

3.2. LMS

The university adopted the Google Workspace for Education product suite[5], which includes Google Classroom (Gclassroom) as the LMS and other utility tools (e.g. Meet) that support online classroom operations. Fig. 2 shows the Gclassroom's class roster of the student class SS207. Each student name is prefixed with the name of a *course group* (e.g. "`CNTT−2C−18`"), to which the student belongs and stays with the student throughout the program. For privacy reason, only the course group prefixes are shown in Fig. 2, the student names are hidden.

3.3. VCS

We chose GitHub classroom as the VCS because it is based on GitHub – a popular distributed VCS – and that it supports Gclassroom integration. The integration is enabled via the class roster feature, such that students are identified on GitHub by their accounts on Gclassroom. The teacher divided students into groups and asked the group leaders to create their assignment groups in GitHub classroom when they join. Other group members simply chose the right group to join. All group members then had access to a GitHub group's workspace, which allows them to create shared repositories and manage projects. The LHS of Fig. 3 shows the GitHub Classroom's roster for class SS207. It was imported from the Gclassroom's roster shown in Fig. 2. Again, for privacy reasons only the course group name prefixes are shown, the student names are hidden. The RHS of the figure shows 9 student groups that were created for the project assignment of the course. For administrative reasons, group G1 had to be divided into two smaller groups midway through the semester to create group G1A. The original and new groups each had 2 students. Other groups had 3 or 4 students.

3.4. jDomainApp Software Framework

The software framework used in the SS2 course is a custom-made Java software framework, named JDo-

[5] https://edu.google.com/products/workspace-for-education

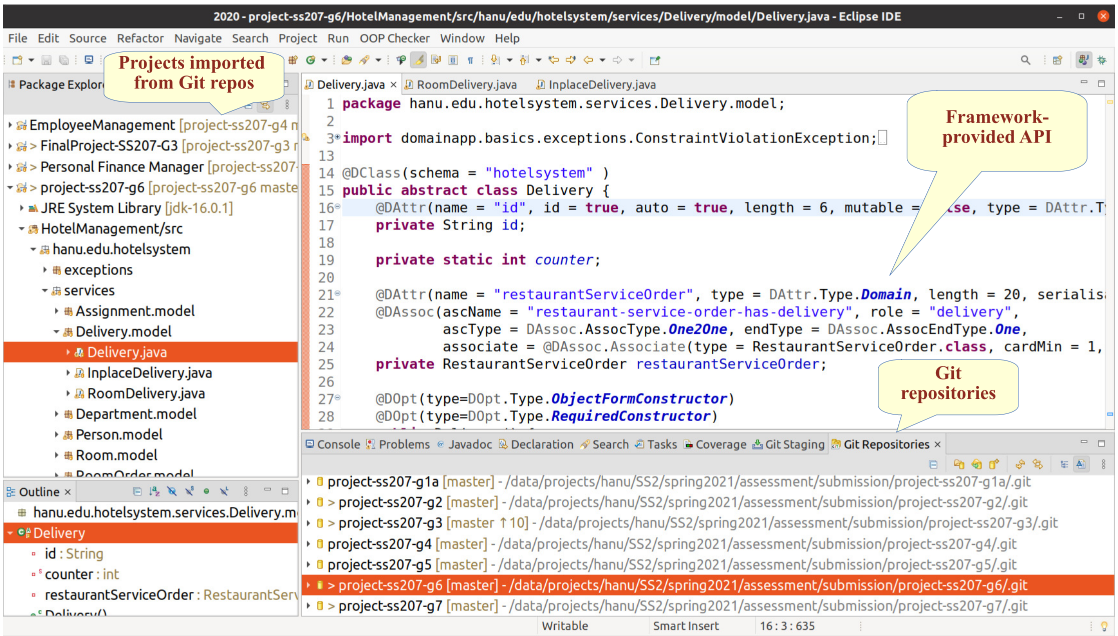

Figure 4: Developing a software using JDOMAINAPP and the Eclipse IDE: the Git repository name of each student group (e.g. "`project−ss207−g6`" highlighted in the "Git Repositories" tab) matches the group name shown in Fig. 3.

MAINAPP, that we developed in previous work [11, 12, 9]. We developed this framework to bring the DDD philosophy [5] into teaching and as a testbed for new research ideas.

JDOMAINAPP adopts the MVC software architecture [3] and supports a module-based design. Each module is constructed as an MVC module that wraps around a domain class in the domain model. JDOMAINAPP's automation capability allows students to focus on developing just the domain model. A basic GUI-based software (including a Swing UI and a database model) is generated and executed automatically. This helps significantly shorten the development cycle, freeing students from the burden of constructing a software from the domain model and, thus, allowing them to concentrate on designing and testing the model. Fig. 4 shows a screenshot of the Eclipse IDE that demonstrates how JDOMAINAPP was used to write a domain class in a student code project. The domain class in question is named `Delivery`, which is part of a model describing a Hotel Management application domain. The code project was imported from the group's Git repository (listed at the bottom of the figure) and is presented on the "Package Explorer" tab. The Java code of the `Delivery` class shows the following three core Java annotations that are provided by the JDOMAINAPP software framework: `DAttr`, `DAssoc` and `DOpt`. The former two annotations are used to describe an associative domain field, while the third annotation describes a domain operation.

3.5. Lessons Learnt

In this section, we briefly reflect on some of the lessons learnt from our delivery of the SS2 course reported above. The first three lessons demonstrate a strong support relationship between SDM and IDM in a PBSE course and how the SDM leverage can further be explored to enhance the IDM of the course. Although the lessons' impacts would be stronger in the online mode, the lessons are helpful for improving the course design in general.

1. Student performance and feedbacks after each SDM's iteration in the software project provide a key input for the IDM's evaluate phase to assess (and, if needed, make adjustments to) the learning objectives and teaching plan.

2. LMS-VCS integration is essential for online PBSE course.

3. GitHub's repository statistics (such as project

contributors[6]) could be used to track and determine a student's code contribution (and individual mark) in the project.

4. Pre-recorded videos of code demonstration would be provided to students to complement the coursebook. This is reported in [13] as being effective for SE courses.

5. Automation capability of software framework plays an important role in enhancing the students experience with the SDM. The automation scope needs to at least cover (1) domain modelling, (2) design model verification and (3) software testing (test case construction and testing automation). These are part of our on-going research with jDomainApp.

6. Project requirements would be made more challenging (and thus differentiate student groups) by including framework-related problems. These concern solutions that overcome existing limitations the framework.

7. Problem context authenticity would be enhanced with the use of the "live" clients, who provide actual user requirements for the software.

4. Conclusion

In this paper, we discussed a concrete instructional design model (IDM) of online PBSE course. The IDM is based on a well-founded instructional design model and is extended with an iterative software development model (SDM). SDM incorporates a software framework to help students practice software development more effectively through its software construction automation capabilities. Another key feature of the IDM is an LMS that supports the integration of a version control system. This system is needed for effective online and collaborative software artefact management. We discussed our experience in applying the IDM to an advanced PBSE course that was conducted during a recent Covid-19 lockdown in Hanoi, Vietnam. The lessons learnt point to a strong support relationship between the SDM and the IDM and that the SDM leverage should be explored further to improve the course. It is unfortutenate that, by the time that we wrote this paper the course students had moved on to other courses and we were not able to contact them for feedbacks. Our plan for future work, therefore, is to conduct a formal implementation of the IDM in a PBSE course and to obtain and analyse the student feedbacks about its effectiveness.

[6]https://docs.github.com/en/repositories/viewing-activity-and-data-for-your-repository/viewing-a-projects-contributors

Acknowledgment

The author would like to thank the Faculty of IT at Hanoi University for their support during the delivery of the SS2 course. The author wishes to thank all students of the SS207 class for their patience and active participation in the course, especially during the Covid-19 lockdown.

References

[1] T. Kanij, J. Grundy, Adapting Teaching of a Software Engineering Service Course Due to COVID-19, in: Proc. IEEE 32nd Conf on Software Engineering Education and Training (CSEE T), 2020, pp. 1–6. doi:10.1109/CSEET49119.2020.9206204, iSSN: 2377-570X.

[2] M. Barr, S. W. Nabir, D. Somerville, Online Delivery of Intensive Software Engineering Education during the COVID-19 Pandemic, in: Proc. IEEE 32nd Conf on Software Engineering Education and Training (CSEE T), 2020, pp. 1–6. doi:10.1109/CSEET49119.2020.9206196, iSSN: 2377-570X.

[3] I. Sommerville, Software Engineering, 10th ed., Pearson, Boston, 2015.

[4] R. A. Reiser, J. V. Dempsey, Trends and Issues in Instructional Design and Technology, 4th ed., Pearson Education, 2017.

[5] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2004.

[6] S. Millett, N. Tune, Patterns, Principles, and Practices of Domain-Driven Design, John Wiley & Sons, 2015.

[7] Apache-S.F, Apache Isis, 2021. URL: http://isis.apache.org/.

[8] Javier Paniza, OpenXava, 2021. URL: http://openxava.org/.

[9] D. M. Le, D.-H. Dang, H. T. Vu, jDomainApp: A Module-Based Domain-Driven Software Framework, in: Proc. 10th Int. Symp. on Information and Communication Technology (SOICT), ACM, 2019, pp. 399–406.

[10] N. Woodthorpe, Use GitHub Classroom with Canvas, Google Classroom, or your own tools, 2019. URL: https://github.blog/2019-08-13-use-github-classroom-with-your-own-tools/.

[11] D. M. Le, D.-H. Dang, V.-H. Nguyen, On Domain Driven Design Using Annotation-Based Domain Specific Language, Computer Languages, Systems & Structures 54 (2018) 199–235.

[12] D. M. Le, D.-H. Dang, V.-H. Nguyen, Generative Software Module Development for Domain-Driven Design with Annotation-Based Domain Specific Language, Information and Software Technology 120 (2020) 106–239.

[13] E. Choi, Applying Inverted Classroom to Software Engineering Education, IJEEEE 3 (2013). doi:10.7763/IJEEEE.2013.V3.205.