Teaching software processes from different application domains

Carla Bezerra
Federal University of Ceará
Quixadá, Ceará, Brazil
carlailane@ufc.br

Emanuel Coutinho
Federal University of Ceará
Quixadá, Ceará, Brazil
emanuel.coutinho@ufc.br

ABSTRACT

In a current application development scenario in different environments, technologies and contexts, such as IoT, Blockchain, Machine Learning and Cloud Computing, there is a need for particular solutions for domain-specific software development processes. The proper definition of software processes requires understanding the involved teams and organization's particularities and specialized technical knowledge in Software Engineering. Although it is an essential part of Software Engineering, many university curricula do not dedicate as much effort to teach software processes, focusing more on the basic principles of Software Engineering, such as requirements, architecture and programming languages. Another important aspect of software processes is modeling. The modeling of a software process provides a basis for managing, automating and supporting the software process improvement. In this context, teaching software process modeling becomes challenging, mainly due to the great emphasis on theory and few practices. This work presents an experience report teaching the definition and modeling of software processes in different domains. We applied in the discipline of software processes a practice for defining and modeling processes in various application domains, such as: IoT, cloud, mobile, critical systems, self-adaptive systems and games. The processes were modeled in the Eclipse Process Framework (EPF) Composer tool based on references from the literature for each domain. In the end, we evaluated the process modeling practice with the students. We concluded that the modeling tool and the maturity in the domain are essential for the good performance of the process.

CCS CONCEPTS

• **Software and its engineering → Software development process management**; • **Applied computing → Education**.

KEYWORDS

software process, systems domain, education.

ACM Reference Format:
Carla Bezerra and Emanuel Coutinho. 2022. Teaching software processes from different application domains. In *XXI Brazilian Symposium on Software Quality (SBQS '22), November 7–10, 2022, Curitiba, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3571473.3571488

1 INTRODUCTION

With the rapid advancement of technologies and computing, the importance of Software Engineering in everyday life is increasing, affecting all aspects of our lives today, including work, learning and education [2]. Software development activity is generally supported by international standards that provide a set of software processes to cover the entire software life cycle and define the activities necessary to design, develop, deploy and maintain a software system, product or service [4].

The need for particular solutions arises in this application development scenario in different environments, technologies and contexts, together with different paradigms. Examples of situations are application development for games [33], machine learning [21], blockchain [36] and self-adaptive systems [3].

A software process can be seen as the set of activities, methods, practices and transformations that guide people in the production of software [30]. The proper definition of software processes requires understanding the particularities of the teams and organizations that will use them, in addition to specialized technical knowledge in Software Engineering [30]. The software process is related to defining the software life cycle, evaluating and improving the software process, and measuring software and software engineering process tools. The software process is inherent in the software practice [23]. Thus, software development teams use, formally or informally, a process to perform the tasks that will culminate in the final software product [30].

Software process models are a way to keep organizations, projects and people together [26]. Therefore, developing, maintaining and improving a software process model are challenging tasks requiring well-trained and experienced process engineers. Although it is an essential part of Software Engineering, most university curricula consider that software processes are on the sidelines of Software Engineering [26]. Typically, these curricula contain classes and labs covering software engineering's basic principles, such as requirements, architecture, and programming languages.

Another important aspect of Software Engineering is the models. Models are built to provide a better understanding of systems or environments [13]. Also, models are important because they allow the representation of ideas, allowing analysis and comparison [12]. However, no model is sufficient and can be analyzed from different perspectives. Process models also become necessary in this context to assist the software process.

The modeling of software processes is an important area of Software Engineering because it provides a basis for managing, automating and supporting the improvement of software processes [8]. Teaching software process modeling is challenging, mainly because it emphasizes theory and offers few practical exercises [8].

Software process education is an important field within software engineering that requires a more practical and realistic approach to learning and teaching [4, 15]. Several studies reported their experiences in learning and teaching software process models and described recommendations and encountered challenges [4–6, 15, 26].

Considering the context described, teachers and students have faced great difficulty concerning the teaching-learning approach to the use of processes in the discipline of Software Engineering, since most of the methodologies adopted today are based on expository classes with little efficiency [35]. Also, the modeling of software processes is often complicated to be applied in the disciplines in an applied manner.

This work presents the experience report of teaching the definition and modeling of software processes in this context. Processes were defined and modeled for different software domains as a practical result of a software process discipline. In addition, to presenting a methodology for teaching software processes, the work, focusing on the definition and modeling of processes, also presents possible suggestions for improvement that can be incorporated into the discipline. Also, examples of processes from different domains that can be used to evaluate and/or execute the process in other disciplines are also available. An evaluation was carried out with the students of the discipline, which we analyzed quantitatively and qualitatively. The main results of the evaluation are that the tool for modeling and maturity in the domain are essential for the good performance of the process.

The rest of the paper is structured as follows. In Section 2, the fundamental concepts used in this work are presented, in addition to the related work. Section 3 presents the work methodology. Section 4 describes the execution of the work. Section 5 shows the evaluation of the teaching methodology. In Section 6, analyzes and discussions about the results are carried out, followed by the limitations of the work in Section 7. Finally, Section 8 presents the conclusions and future work.

2 BACKGROUND

2.1 Software Process Modeling

Software process modeling is a well-known topic in Software Engineering research studied for the past twenty-five years. Several process modeling languages have been proposed over the years [20]. Among them, the most evolved was SPEM (Software & Systems Process Engineering Metamodel), which is the standard notation of OMG (Object Management Group) for modeling software development processes [31].

Figure 1 presents the elements of the SPEM notation. The content of the method is expressed mainly using work product definitions, role definitions, task definitions and guidance. Guidance, such as guidelines, checklists, examples or roadmaps, is defined at the intersection of the content of the method and the process. On the right side of the diagram, the elements used to represent processes in SPEM 2.0 are highlighted. The main element is the activity that can be nested to define division structures and related to each other to define a workflow. Activities are used to define processes [31].

One tool that supports SPEM 2.0 notation is Eclipse Process Framework Composer (EPF Composer)[1]. It is an open-source tool aiming to support customizable software processes' modeling. An advantage of this tool, in addition to following a standardized notation, is the modeling of more robust processes that can be made available from a publication of the process. In this work, we used the EPF Composer tool for teaching software process modeling, based on several models and frameworks to support the definition of processes in several application domains.

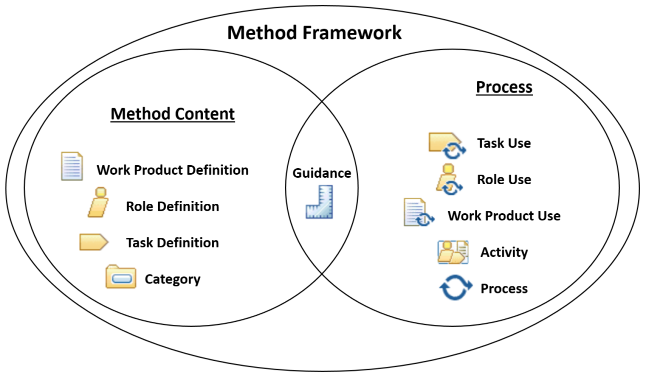

Figure 1: Key terminology defined in this specification mapped to Method Content versus Process (based on [31])

2.2 Software Domain Processes

The development environment and method play a vital role in the success of software development [37]. Software development processes have evolved over time to meet the changing needs of users and the industry. The adoption of new technologies (such as the advent of big data, cloud computing and IoT) is another reason for the evolution of software development processes.

In the literature, there are several processes proposed for different domains, such as: games [2, 32], internet of things [37], mobile applications [18, 22], self-adaptive systems [3], machine learning systems [28], among others. However, there is a need to adapt the processes to the characteristics of these domains. For some types of systems, the delay inherent in traditional change processes is unsatisfactory [3], as they are often critical systems that need to operate continuously. In traditional change processes, changes are implemented during scheduled downtime and, as a consequence, continuous operation is not possible.

Osborne O'Hagan et al. [32] described a systematic review of the literature on software processes used in game development. Various process models used in industry and academia and research were presented, with discussions on software process improvement initiatives for game development. Factors that promote or prevent the adoption of process models and their implementation were highlighted. The results indicated that no single model serves as a model for the process of best practices for the development of games, deciding which model is more suitable for a specific game. Agile models like Scrum and XP are suitable for the domain of game development, where innovation and speed to the market are vital. Hybrid approaches, such as reuse, may also be suitable for game development. The initial investment risk in terms of time and cost

[1]EPF Composer - https://www.eclipse.org/epf/composer_architecture/

is mitigated with a game with stable requirements and a longer lifespan.

The multidisciplinary nature of game development processes that combine sound, art, control systems, artificial intelligence (AI) and human factors make the development of software games different from traditional software development. Software engineering techniques help game development to achieve maintainability, flexibility, less effort and cost, and better design. Aleem et al. [2] assessed the state of the art of research on the game development software engineering process, highlighting areas that need to be considered by researchers through a systematic literature review. The largest number of studies were reported in the production phase of the game development software engineering process, followed by the pre-production phase. In contrast, the post-production phase received far less research activity than the pre-production and production phases. This study suggests that the game development software engineering process has many aspects that require more attention, especially the post-production phase.

Cloud computing helps to reduce costs, allows scalability and improves communication through its services. Younas et al. [37] evaluated a generic framework combining agile development and cloud computing with a case study. Before conducting the case study, participants were trained in the framework. The case study results show that the performance of agile methods is improved using the framework. The improvement is measured in terms of local and distributed agile development environments. Also highlighted was the range of tools to support agile development activities in the context of cloud computing and its compatibility.

Fontão et al. [18] presented MSECO-DEV, a process to support external developers in achieving central organization goals by developing mobile applications. MSECO-DEV comprises 8 activities, 7 artifacts, 8 recommendations and 17 practices. Activities, recommendations and practices were evaluated by 65 Brazilian developers who worked with several MSECOs (Mobile Software Ecosystem) to assess their benefits for the routine of developing mobile applications. As a result, it was found that developers have difficulty carrying out marketing activities and finding support materials for development. Practices, activities and recommendations were also evolved and adjusted to define the MSECO-DEV.

Andersson et al. [3] discussed for self-adaptive systems how some activities that traditionally occur at development time are moved to runtime. Responsibilities for these activities shift from software engineers to the system itself, blurring the traditional boundary between development and runtime. Consequently, we argue how the traditional software engineering process needs to be redesigned to distinguish between runtime and runtime activities and support designers in making decisions about designing these systems properly. Several challenges related to this necessary reconceptualization were identified and initial ideas based on process modeling were proposed. The SPEM notation was used to specify which activities should be performed offline and online and their dependencies.

Liu et al. [28] performed a qualitative interview study to uncover emerging tasks in development processes when machine learning components are used within software systems. The study identified 25 software development tasks for these systems. However, the process for mastering machine learning systems was not developed at work.

Liu et al. [29] conducted a qualitative study from interviews to discover tasks in the software development processor used to build machine presentation systems. The study identified 25 different tasks that can contribute to developing a specific process for this domain.

2.3 Teaching Software Processes

The teaching of software processes has been discussed in some works identified in the literature in recent years [6, 8, 16, 26, 30].

Kuhrmann et al. [26] proposed the inclusion of software processes more explicitly in Software Engineering curricula. For this, a course at the master's level was designed and implemented in which students learn why software processes are needed and how they can be analyzed, designed, implemented and improved. The course structure, objectives and corresponding teaching methods were also presented. The lack of problems to effectively prepare students for the industry was discussed, and the lack of education in software and modeling processes was identified as a major deficiency.

Fernandes et al. [16] presented an empirical study to assess how online resources support the software learning process in an online Software Engineering course with video lessons, online questionnaires and discussion forums. The results showed that videos and online questionnaires contribute to the improvement of up to 15% of students' grades in software process questions compared to students who do not watch videos or answer online questionnaires. However, based on two exam questions that were repeated over the three years, it was found that the improvement in grade seems to be related mainly to the video classes attended, instead of questionnaires answered online.

Calderón et al. [6] described state of art related to serious games for education in software process patterns to identify current games in terms of scope, main features and perceived benefits of integrating them in education software processes, in addition to identifying research opportunities. The study was conducted as a multivocal literature review that follows a pre-defined procedure in which scientific and gray literature studies are analyzed. The results revealed that serious games have the potential as support tools for teaching software process patterns. Still, more research and experimental results are needed to see the full potential as learning resources.

Chaves et al. [8] described a formal experiment carried out to assess the learning effectiveness of a serious game (DesigMPS), designed to support the teaching of software process modeling and to compare game-based learning with a method of project-based learning. In the game, the student models a software process from the perspective of software process improvement, based on the Brazilian model (MPS.Br). The results indicate that playing the game can have a positive learning effect and results in a greater degree of learning effectiveness than the project-based instructional learning method.

Moura and Santos [30] presented an educational game (ProcSoft) with the objective of teaching concepts, definition, structure and content of a software process, and good software engineering practices in an informal and relaxed way. Students learn to compose a process more completely in the game about the software development life cycle phases. The basis of the game was the ISO/IEC 29110 standard, which describes a software process composed of activities from the basic development and project management cycle at a high level. The results showed the involvement of the participants in the classroom, the contribution to learning and the positive influence in the search for additional knowledge.

In general, these works seek to highlight the importance of software development processes, with experiences in different approaches, applied in the classroom with students. We highlighted the use of serious games [6, 16, 30] for teaching software development processes. The proposed work also sought to promote software development processes in teaching, with the additional feature of defining and modeling a process based on SPEM notation.

3 METHODOLOGY

This section describes the planned and applied methodology for the Software Processes discipline for teaching the definition and modeling of software processes in different domains. The discipline in question belongs to the fifth semester of the undergraduate course in Software Engineering. The main goal of this methodology is the application of the concepts learned in the discipline, adapting the process to the specific characteristics of a particular software domain.

This work will follow three steps to meet the proposed goal: planning, execution and analysis. Figure 2 illustrates the three steps and the sequence of activities. The details of each step are described below.

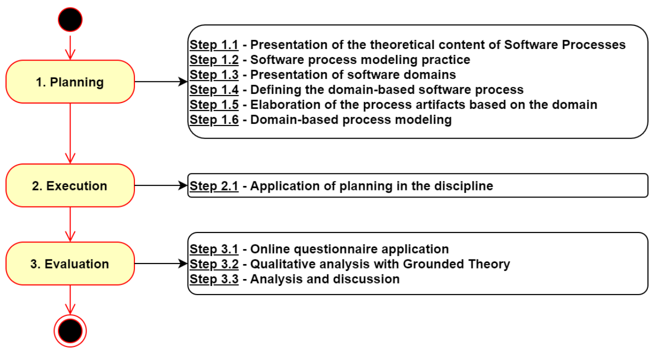
1. Planning
Step 1.1 - Presentation of the theoretical content of Software Processes
Step 1.2 - Software process modeling practice
Step 1.3 - Presentation of software domains
Step 1.4 - Defining the domain-based software process
Step 1.5 - Elaboration of the process artifacts based on the domain
Step 1.6 - Domain-based process modeling
2. Execution
Step 2.1 - Application of planning in the discipline
3. Evaluation
Step 3.1 - Online questionnaire application
Step 3.2 - Qualitative analysis with Grounded Theory
Step 3.3 - Analysis and discussion

Figure 2: Representation of the work methodology

In Figure 2, the first phase involves planning the discipline. The planning is divided into the following 6 steps: (1) presentation of the theoretical content of processes. (2) practice of modeling software processes. (3) presentation of software domains. (4) defining the domain-based software process. (5) elaboration of process artifacts based on the domain. (6) domain-based process modeling.

The second phase of the methodology consists of executing the discipline's planning. At this time, all activities that make up the planning step are carried out with the students.

The third phase consists of evaluating the results, and includes three steps: (1) application of the online questionnaire; (2) qualitative analysis with Grounded Theory; and (3) analysis and discussion.

For qualitative analysis, this work used procedures from the Grounded Theory [11] methodology, inspired by the analysis approach presented in Ferreira et al. [17]. The Grounded Theory aims to create a theory from the data collected and analyzed systematically, consisting of three steps: (1) open coding, (2) axial coding and (3) selective coding. In open coding, data is broken, analyzed, compared, conceptualized and categorized [11]. In axial coding, categories are associated with subcategories, forming more related and dense categories. Finally, in selective coding, the central category or idea of the study is identified, corresponding to the theory in which all categories are related. Strauss and Corbin also point out that the researcher can use only a few steps to achieve his research objective [11]. Therefore, in this research, only steps 1 and 2 of Grounded Theory were used to identify the categories and their relationships. Additionally, as a way to avoid trends in the analysis, another researcher reviewed the result.

4 REPORTING OF SOFTWARE PROCESS TEACHING

This section presents a report on the Software Processes discipline, describing the planning, execution and analysis strategy in the next subsections.

4.1 Discipline Planning

Step 1 - Presentation of the theoretical content of Software Processes:

Initially, the idea is to present the content covered in the discipline of Software Processes. The content consists of process definitions, life cycle, traditional process methodologies, maturity models, agile methodologies and process modeling languages. Table 1 presents the theoretical content taught for teaching processes. It is worth mentioning that the basic bibliography for teaching software processes is limited. There is no base book with all the content of processes. The base books used in this discipline are the Software Engineering books of Sommerville [34] and the Software Quality book of Koscianski [24]. However, not all the course content is presented in these books, requiring additional materials available on the web.

Table 1: Theoretical content of the Software Processes discipline

ID	Content
1	Introduction to software processes
2	Process framework: RUP
3	Agile methodology: Scrum
4	Agile methodology: Extreme Programming (XP)
5	ISO standards: ISO 12207
6	ISO standards: ISO 15504
7	Process maturity model: MPS.Br
8	Process maturity model: CMMI
9	Software process modeling

Step 2 - Software process modeling practice:

After the presentation of the content, practical classes on the modeling of software processes are held. The notation used for the course was the SPEM [31], supported by the EPF Composer tool. Other notations could also be used, such as BPMN notation. The choice for modeling using the SPEM notation and the EPF Composer tool was because it was used widely in other disciplines of Software Processes already taught by the teacher and also to support the more robust modeling of the domains.

The practice consists of a practical example of modeling a fictional process following all the steps of the EPF Composer tool until the publication of the process. EPF Composer tutoring is also available so that students can proceed to practical work.

Step 3 - Presentation of software domains:

Six different application domains are presented to the class: self-adaptive systems, mobile applications, applications in the cloud, internet of things (IoT), applications for games and critical systems. Articles referring to the domain were made available for all these domains. Table 2 presents the main contents made available to teams with the main characteristics of the selected domains. All selected articles have Software Engineering activities focused on the specific domain. Some articles present a defined process for the domain and others a systematic review of the literature.

Table 2: Theoretical reference for domains

Domain	Theoretical Reference
Self-Adaptive Systems	[3, 14]
Mobile Applications	[18, 22]
Cloud Applications	[10, 25]
Internet of Things	[27, 33]
Games Applications	[2, 32]
Critical Systems	[1, 7]

Step 4 - Defining the domain-based software process:

In this step, students must define the process based on the domain using some models of processes presented in the discipline. The Table 3 structure is used for definition. In the structure of Table 3, a process comprises two or more subprocesses. Each subprocess has a set of related activities. The defined process must contain the subprocesses of Requirements, Analysis and Design, Implementation, Tests and Project Management.

Table 3: Process definition template

Subprocess 1: Subprocess name	
The purpose of this subprocess is ...	
Activity:	Activity name
Description:	Activity description
Pre-activity:	What was the activity before this activity?
Responsible:	Roles responsible for carrying out the activity
Required Artifacts:	Activity Entry Artifacts
Generated Artifacts:	Activity Output Artifacts
Post-activity:	What is the activity after this activity?
Steps:	What are the necessary steps to perform this activity?

Step 5 - Elaboration of the process artifacts based on the domain:

After defining the process, the main process artifacts generated in the activities are identified. A template is given to students to standardize these artifacts. Students must develop these artifacts based on what is requested in the activities. In addition, artifacts must be adapted to meet the characteristics of the domain.

Step 6 - Domain-based process modeling:

Two weeks of classes were dedicated to teaching process modeling using the EPF Composer tool. Students should model the entire defined process and attach the generated artifacts to the process. In this step, students will better define the process flows. Thus, depending on the refinement of the process flow, there may be changes in the definition of the process and in the artifacts. As a delivery, the final published process is expected and also customized according to the domain.

4.2 Execution

The teaching methodology was applied to the Software Process discipline of the Software Engineering course in the first semester of 2019. The class had 30 students, most of them from the fifth semester. This discipline is mandatory in the Software Engineering course and has as a prerequisite an initial discipline of Introduction to Software Processes and Requirements. In addition, students have already taken courses in software development, Analysis and Design and Systems Development Project. In parallel, the students cursed the disciplines of Requirements, Verification and Validation, Web Development and Mobile Application Development. Thus, students already have some general maturity for Software Engineering activities and also have knowledge of some domains.

The definition and modeling of the domain-specific software process correspond to the discipline's final work. However, due to the complexity of the work, more time is needed in the discipline to develop the work. In this way, the work is developed in about two months with partial deliveries for evolution and feedback on the process by the teacher for the teams.

The work was carried out in teams of 5 to 6 students. The process domains were suggested by the discipline teacher. The team that had more affinity with the theme applied to define the respective process. When more than one team showed interest in the same domain, a division draw was carried out. In addition to the domains suggested in Table 2, the domains of Systems with User-Oriented Development and Embedded Systems were also suggested. There were no teams interested in these domains.

The first delivery of the work consisted of the initial definition of the process based on the template presented in Table 3. The students' teams presented a high-level process, without details of the steps of each activity and without the artifacts. This feedback was important for students to define the process correctly. In this delivery, it was noticed that the students defined many activities. Some activities were unrelated to each other and other activities important to the domain were not included. The low level of experience of most teams in the domain was also noticed.

The second delivery was made with the definition of the process evolved with the steps and the artifacts. At this stage, the students left the steps poorly detailed, and the artifacts often did not reflect the activities or were not specific to the domain. Feedback was also given on the necessary corrections for each process.

After the process was defined and corrected, the third delivery consisted of the process modeled in EPF Composer. Although the tool is complete and meets more robust process modeling, there were problems with the tool reported by students, such as: it only works on the Windows operating system, the configuration management in the tool is complicated, and some features are difficult to understand and the flows visually break frequently. Despite the tool's problems, all the teams were able to model and publish the process.

The fourth delivery consisted of the final presentation of the process by the team reporting the weaknesses, strengths and lessons learned from the work. The score for each partial delivery (the first three deliveries) has a weight of 1 and the final delivery (fourth delivery) has a weight of 2.

The main characteristics of each process defined for the domains are described below:

- **Process for Self-Adaptive Systems:** one of the main adaptations for building systems for the domain was the use of the KAOS model Knowledge Acquisition in automated specification ou Keep All Objectives Satisfied) to specify the objectives of the system, requirements and uncertainties [9]. The MAPE-K model was used in the design phase to build the architecture. In the tests, the process adapts the test case spreadsheet proposed in [19]. In addition, the project management in the process used the Scrum agile methodology.
- **Process for Mobile Applications:** The project management of the process adopts some Scrum practices. The project focuses on the design of the user interface and experience and uses prototyping. Configuration management activities are also used. Testing activities are based on user experience testing across multiple devices. The distribution process is the great differential of the process, based on marketing activities and deploy on app stores.
- **Process for Cloud Applications:** The process for developing cloud applications is based on the Microsoft Azure cloud environment. The process is also based on some practices of eXtreme Programming (XP), such as: refactoring, continuous integration, coding standards and pair programming. Project management is based on some Scrum practices. The configuration management process is entirely adapted to the Azure environment so that applications can be developed in that environment.
- **Process for IoT Applications:** Project management is based on some Scrum practices. The requirements are raised from a specification of the vocabulary of the application domain. The choice of architecture is based on the characteristics of the domain. Mapping is carried out between the architecture specifications and the domain vocabulary. Automated tests are also provided according to the vocabulary of the domain.
- **Process for Games:** The process defined for games was based on the agile Scrum methodology. The definition of the game is built from the document of Game Design, which is the main artifact of the process. The tests performed are on gameplay and usability of the game performed with end users from beta versions of the game.
- **Process for Critical Systems:** Project management is focused on risk management. The project activity developed in the process has an architectural model output. The testing activity was based on the characteristics of critical systems, determining each unsafe state and criticality factors of the system. Agile methodologies were not used to define the process, only the literature on critical systems.

The developed processes are available at the link[2].

5 EVALUATION

For the evaluation of the process modeling methodology for the domains applied in the discipline, an online questionnaire was answered by 15 students of the discipline Software Processes. The questionnaire presented in Table 4 was composed of three types of questions: demographic (1 to 6), about software and tool processes (7 and 8), and opinion (9 to 11). The number of respondents corresponded to only half of the class. We applied the questionnaire at the end of the course; therefore, not all students responded. However, we got answers from all the students representing the 6 teams that developed the processes. We made it very clear that student answers and processes would be used for academic and scientific purposes, with student agreement.

Table 4: Questionnaire applied to students

ID	Question
1	What is your entry semester?
2	What was your team?
3	What is your level of experience in applying your teams?
4	What materials did your team rely on to go deeper into the process domain?
5	How much time did you spend on the process?
6	How much do you think your process can be applied in the real world/market?
7	What methodologies were used to build the process? (you can check more than one option)
8	Was the tool used for modeling the process adequate? What are the positive and negative points of using the tool?
9	What are the strengths of the discipline?
10	What are the weaknesses of the discipline?
11	How could your team's process be improved?

The first part of the questionnaire was made up of demographic questions, to find out the profile of the students who answered the survey and about the effort and dedication to the activities.

As for the semester of entry of students, it was identified that 13 (86.7%) belonged to 2017.1, and the rest, 2 (13.3%), to the 2016.1 semesters. This information is interesting because it highlights that most students are in regular semesters.

Regarding the team, the distribution of students in the six teams was as follows: Processes for self-adaptive systems 2 (13.3%), Processes for mobile applications 3 (20.0%), Processes for IoT applications 4 (26.7%), Processes for critical systems 2 (13.3%), Processes for Games 3 (20.0%), and Processes for development in the cloud 1 (6.7%).

[2]https://doi.org/10.5281/zenodo.7068357

The level of experience in the field of application of the students' team varied, focusing on little and no experience: Very experienced 1 (6.7%), Intermediate experience 2 (13.3%), Little experience 7 (46.7%), and No experience 5 (33.3%).

The materials used by the teams to deepen the domain were diversified. Many reports from searches in scientific articles indicate a certain academic maturity. However, as many subjects have very technical characteristics, websites and blogs were also consulted. Searches for similar applications and processes and process adaptations were cited. Finally, information was also sought in manuals and expert reports.

The time of dedication in the activities varied from 12h to more than 40h. However, most answered around 4h. The time divisions were: 16h - 7 (46.7%),14h - 1 (6.7%),12h - 4 (26.7%),more than 20h - 2 (13.2%), and more than 40h - 1 (6.7%).

Regarding the applicability of the process in the market, among the options students reported that yes, they are applicable, but with reservations: Applicable with few adaptations 5 (33.3%) and Applicable with many adaptations 10 (66.7%).

The second part of the questionnaire consisted of questions about the application of methodologies in activities and the used process modeling tool, in this case, EPF Composer.

Figure 3 shows the methodologies, process models and frameworks used by students in constructing their processes. The student could select more than one option. The highlight for Scrum, was the most mentioned but was not used in all processes.

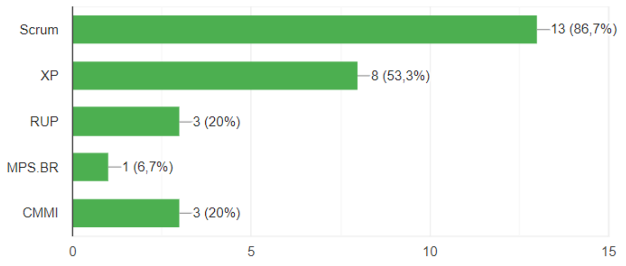

Figure 3: Methodologies, process models and frameworks used to build processes

One of the questions focused on the tool used in the discipline for modeling processes. The students reported on the positive and negative aspects of its use.

In general, in the opinion of the students, the tool fulfills its role, which is to enable the modeling of software development processes. It was considered with many features and details. This avoids much manual work, as the tool registers the process. In addition, it provides different views of the process.

However, many reports of defects have been identified in the tool, in addition to the lack or little documentation and support. There were reports of difficulties with the installation. Also from the point of view of usability, it was considered bad, being confused in several items. Finally, difficulties in its use in teams, cause rework.

Finally, the third and last part of the questionnaire was composed of open questions, with opinions on strengths, weaknesses and improvements in the process. In this analysis of the answers, we used procedures from the Grounded Theory (GT) [11] methodology using the Atlas.ti tool[3]. GT aims to create a theory from the data collected and analyzed systematically, consisting of three phases: (1) open coding, (2) axial coding and (3) selective coding.

In open coding, a break, analysis, comparison, conceptualization and data categorization is performed [11]. In axial coding, categories are associated with their subcategories, forming more related and dense categories. Finally, in selective coding, the central category or the study's idea is identified, corresponding to the theory in which all categories are related. Strauss and Corbin explain that the researcher can use only a few steps to achieve his research goal [11]. So, in this research, we used only the GT's phases 1 and 2 to identify the categories and their relationships. Additionally, another researcher reviewed the analysis to avoid biases.

In this research, 14 categories (codes) were identified, listed in order of decreasing frequency: Standards and Models (9), Process Definition (7), Theory (7), Practice (6), Company (5), Methodology (5), Opinion (5), Modeling (4), Time (4), Development (3), Inexperience (3), Didactic (3), Process Evaluation (2) and Support (2). Table 5 displays the description of each category. Not all were identified in the strengths, weaknesses and improvements simultaneously.

Additionally, 9 relationships between codes were identified: Process definition is associated with Development, Process definition is associated with Standards and Models, Process definition is associated with Time, Inexperience is a cause of Time, Standards and Models is associated with Company, Standards and Models is associated with Development, Practice is associated with Didactics, Time is the cause of Methodology, and Theory is associated with Didactics.

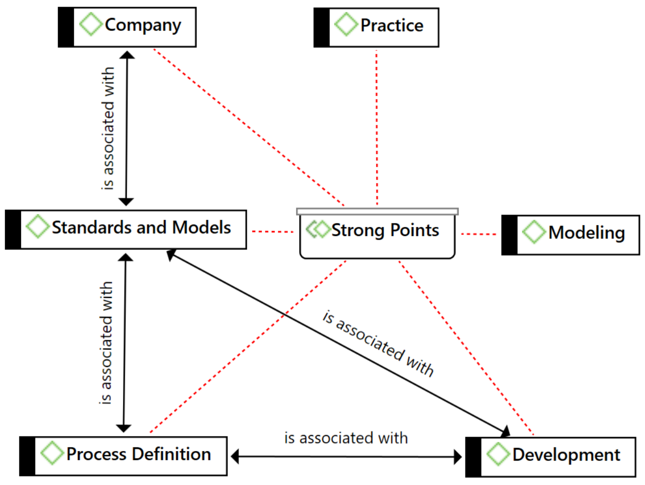

Figure 4: Strong points

Figure 4 shows the strengths identified in the research, and their codes and relationships. The relationship *Standards and Models is associated with Company* described that the relationship between company/software development processes was somewhat recognized by some students, highlighted in P1's response with "*Understanding the processes helps to understand how companies are*

[3]https://atlasti.com/

Table 5: Description of the categories identified in the qualitative analysis

Category	Description
Development	Application development (requirements, analysis, design, implementation, testing)
Didactics	Teacher's actions in the classroom, teaching strategies, ways to approach content
Company	Mentions on the business environment, professions, impacts outside the classroom
Inexperience	Lack of experience in software processes and prerequisites
Methodology	Methodology of the discipline, as planned, forms of evaluation, activities
Modeling	The process model itself, images, representations, how to develop a process model
Process Definition	Structure, sequence, dependencies, the project itself, activities
Process Evaluation	Elements of using the process, testing and evaluating the results
Standards and Models	Documents, guides, standards
Opinion	Opinions and feelings of students about the discipline
Practice	Practical activities or practice in the development process
Support	Support in the discipline in the development process and in the tool adopted for modeling
Time	Duration of activities in the discipline
Theory	Theory behind the subjects covered, theoretical components of the discipline

engaged with software development. This knowledge contributes not only to the development itself, but to a better understanding of how companies and developers are contributing to improving software and the entire development environment.". This is important because given the theoretical nature of the discipline, a view of importance for companies is a benefit. In the relationship *Standards and Models are associated with Development* it was noticed by some answers that standards, guides and models have an important role in the development of applications, reported by P2 in "*In the discipline it is possible to explore the in-depth knowledge about different processes, moreover, it is necessary to have a defined process, so that software production will be more effective and with less risk of future problems*". In the relationship *Process Definition is associated with Standards and Models*, in some responses the relationship between standards and the development of software development processes as a basis was verified, enabling a better result, highlighted by P3 with "*The course allowed to learn a general overview about the content of software processes. It was possible to know how a software process is conceived and created*". Likewise, the relationship *Process Definition is associated with Development* showed a good relationship between process definition and development, reported by P15 in "*Helps to better understand the structure of a process and its importance in development*".

Figure 5 shows the weaknesses identified in the research, with their codes and relationships. The relationship *Theory is associated with Didactics* emerged as an observation in relation to the theoretical load of the discipline, which can harm the didactics of the teacher, as reported by P1 in "*Quite theoretical in some periods and not very dynamic concerning the transmission of content*". The *Time is cause of Methodology* relationship, on the other hand, was the highlight on the duration of activities that could be shorter, thus the time better distributed with other tasks of the discipline, reported by P7 in "*In my point of view, an exaggerated time was spent to present the individual articles, time that could be used to monitor the construction of the process better*". In the relationship *Practice is associated with Didactics*, it was realized the need to carry out more practical activities in the discipline, and to be more dynamic or with more appropriate didactics, according to the observation of P13 with "*A more practical approach was lacking, for example, use of games to use in explaining some processes*".

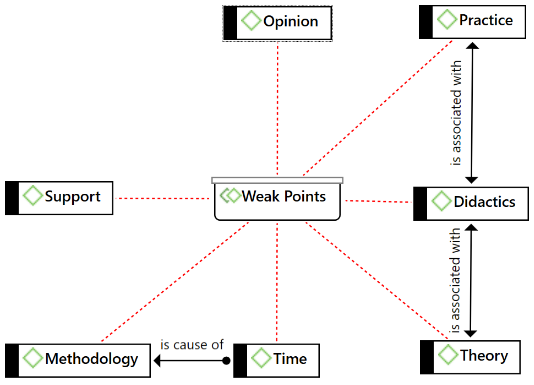

Figure 5: Weak points

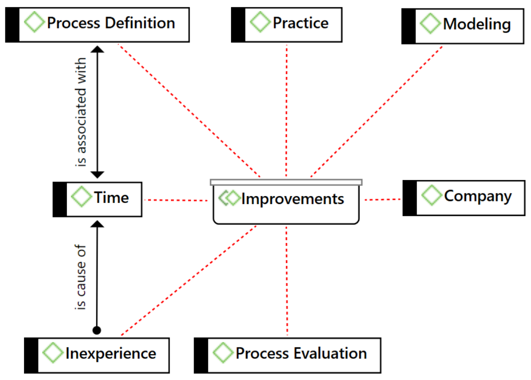

Figure 6: Improvements

Figure 6 shows the codes and relationships identified in relation to the improvements. The relationship *Process Definition is associated with Time* highlighted the level of detail that the process developed in the discipline could have, but the workload would not be enough, besides the effort. This aspect was highlighted by P6 in "*It could be more in-depth and rich in details with a longer research that would not fit the scope of the discipline*". The relationship *Inexperience is a cause of Time* highlighted the issue of the importance of experience in the development of a software development process. The inexperience in the domain in which the process will be addressed was commented by some students. The inexperience in software development processes also contributes to the rework, as reported by P8 in "*We were unable to finish the whole process due to lack of time, we lost our work about 3 or 4 times and we had to redo it. And I think there are still many things to improve, our research source was not specific enough to do everything that a robust process needs, in addition to not having experience*".

6 DISCUSSION

Figure 3 shows the methodologies, process models and frameworks used by students in constructing their processes (Scrum, XP, RUP, MPS.Br, and CMMI). They can be used in teaching software development processes, with the necessary adaptations. An important factor highlighted in the question about the tool for teaching software development processes, and the qualitative analysis was the correct tool selection. It must be adequate and have good usability and documentation. Thus, it is possible to avoid rework with modeling.

As great results of the qualitative analysis we obtained: (i) the importance highlighted by students of standards and models for professional life and the development of applications; (ii) didactics in software processes have a great influence on students in teaching software development processes, and if theory and practice are well balanced, learning also improves; and (iii) the definition of the process (structure and modeling) requires effort and dedication.

The entrance semester of students reveals that they are in the correct period of the course, the correct semester of the discipline, and have passed the prerequisites. It is worth reflecting on their performance in this process discipline as a consequence of the prerequisites. In general, disciplines of analysis and design, requirements and development have already been taken, and such knowledge is very useful for the discipline of the software development process. However, the maturity in development did not reflect so much in the experience in the domain, where most responses (12 out of 15) responded with little or no experience in the domain. This directly impacts the elaboration of the process.

Regarding the applicability of the process in the market, everyone responded that it exists with adaptations. This is corroborated by the qualitative analysis, which highlighted the relationship of standards and models with companies as strengths. This relationship can be evidenced by the P7 speech with "*Can give general visibility about the processes, and shows the importance and facilities that a high level of process in a certified organization can bring to the development team, business, among other departments of an organization. Another point is the placement of the practice of building a process*".

Regarding the processes elaborated in the discipline, other disciplines can benefit from the processes or experiments, such as Cloud Computing, Game Development, and Mobile Application Development. However, the processes need adaptations according to the methodology applied in the disciplines.

In future work, we intend to explore other process methodologies and use other modeling notations to compare the methodology results. In addition, we also intend to suggest the development of processes from other domains such as blockchain and machine learning.

7 THREATS TO VALIDITY AND RESEARCH LIMITATIONS

One of the limitations of the work was only half of the class (15 students) answered the questionnaire. However, they had representatives from all 6 defined processes. It is expected that the methodology will be used in other classes of Software Processes. Because few students answered the questionnaire, few responses were obtained, both quantitative and qualitative. For qualitative analysis, the results could have differed if there were more answers with texts describing situations and opinions. Application in more than one class is also necessary for greater depth in the results.

The fact that it was applied to only one class of Software Processes threatens the validity and a limitation of the work because the context of the moment may have influenced the responses. The idea was to apply in different classes and semesters for a better response.

Regarding the use of the EPF Composer process modeling tool, several students reported that its lack of support and defects hindered the performance of the project and the modeling itself. Tool training can minimize these losses or the help of a monitor in the discipline for support. In addition, the possibility for other process modeling notations most used in the industry, such as business process management notation (BPMN), is supported by tools such as Heflo[4].

Another limitation is that the subject teacher only rated the quality process following the methodologies used. Ideally, people experienced in the domains would evaluate each process with the teacher. Furthermore, the evaluation focused on the discipline, not the methodology to define processes for different domains. We will apply the methodology in other disciplines to cover the evaluation process definition methodology for different domains.

Finally, inexperience in programming and software development processes also impairs performance and results in the discipline. One way to improve performance in this aspect would be in prerequisite disciplines if more software development processes are discussed, eventually, design models to organize the development, inserting the importance of software processes in programming disciplines. Thus, the culture of processes would already be diluted in other disciplines, being deepened in software processes.

8 CONCLUSION

This paper describes an experience of teaching software development processes. The proposed approach was applied in the Software Process discipline of an undergraduate course. The contribution of this work was: (i) an approach to teaching software processes for different domains (self-adaptive systems, mobile applications, cloud applications, IoT systems, games applications and critical systems); (ii) discussions on the use of the EPF Composer tool in

[4]https://www.heflo.com/pt-br/

process modeling; and (iii) a qualitative analysis with Grounded Theory on the results of an online questionnaire.

This work benefits students and teachers who work directly with software development processes, mainly for non-traditional processes. Its application is directly associated with classes and process modeling. For software quality researchers, it is also possible to investigate the effects of software development processes in non-traditional areas and their effects in practice. Future work intends to: (i) apply the methodology in other classes to obtain more reliable results; (ii) measure the performance of the applied methodology from the viewpoint of learning software development processes; (iii) use other process modeling notations such as BPMN; and, (iv) use other domains for process modeling such as blockchain and machine learning.

REFERENCES

[1] A. A. Abdelaziz, Y. El-Tahir, and R. Osman. 2015. Adaptive Software Development for developing safety critical software. In *2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE)*. 41–46.

[2] Saiqa Aleem, Luiz Fernando Capretz, and Faheem Ahmed. 2016. Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development* 4, 1 (2016), 6.

[3] Jesper Andersson, Luciano Baresi, Nelly Bencomo, Rogério de Lemos, Alessandra Gorla, Paola Inverardi, and Thomas Vogel. 2013. *Software Engineering Processes for Self-Adaptive Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 51–75. https://doi.org/10.1007/978-3-642-35813-5_3

[4] Alejandro Calderón, Manuel Trinidad, Mercedes Ruiz, and Rory V O'Connor. 2018. Teaching software processes and standards: a review of serious games approaches. In *International Conference on Software Process Improvement and Capability Determination*. Springer, 154–166.

[5] Alejandro Calderón, Manuel Trinidad, Mercedes Ruiz, and Rory V O'Connor. 2019. An Experience of Use a Serious Game for Teaching Software Process Improvement. In *European Conference on Software Process Improvement*. Springer, 249–259.

[6] Alejandro Calderón, Mercedes Ruiz, and Rory V. O'Connor. 2018. A multivocal literature review on serious games for software process standards education. *Computer Standards & Interfaces* 57 (2018), 36 – 48.

[7] Gabriella Carrozza, Roberto Pietrantuono, and Stefano Russo. 2018. A software quality framework for large-scale mission-critical systems engineering. *Information and Software Technology* 102 (2018), 100–116.

[8] R. O. Chaves, C. G. von Wangenheim, J. C. C. Furtado, S. R. B. Oliveira, A. Santos, and E. L. Favero. 2015. Experimental Evaluation of a Serious Game for Teaching Software Process Modeling. *IEEE Transactions on Education* 58, 4 (Nov 2015), 289–296. https://doi.org/10.1109/TE.2015.2411573

[9] Betty HC Cheng, Pete Sawyer, Nelly Bencomo, and Jon Whittle. 2009. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 468–483.

[10] Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C Gall. 2015. The making of cloud applications: An empirical study on software development for the cloud. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 393–403.

[11] Juliet Corbin and Anselm Strauss. 2014. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications.

[12] Emanuel Coutinho, Carla Ilane Moreira Bezerra, Leonardo Oliveira Moreira, and Igor Rodrigues. [n.d.]. A Tool for Software Ecosystem Models: An Analysis on their Implications in Education. In *34th Brazilian Symposium on Software Engineering (SBES '20)* (Natal, Brazil) *(SBES 2020)*. ACM, New York, NY, USA. https://doi.org/10.1145/3422392.3422486

[13] Emanuel Ferreira Coutinho, Davi Viana, and Rodrigo Pereira dos Santos. 2017. An Exploratory Study on the Need for Modeling Software Ecosystems: The Case of SOLAR SECO. In *9th International Workshop on Modelling in Software Engineering (MISE)* (Buenos Aires, Argentina) *(MISE '17)*. IEEE Press, Piscataway, NJ, USA, 47–53. https://doi.org/10.1109/MiSE.2017.3

[14] Rogério De Lemos, Holger Giese, Hausi A Müller, Mary Shaw, Jesper Andersson, Marin Litoiu, Bradley Schmerl, Gabriel Tamura, Norha M Villegas, Thomas Vogel, et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*. Springer, 1–32.

[15] Jose; Augusto De Sena Quaresma and Sandro Ronaldo Bezerra Oliveira. 2021. Teaching and learning strategies for software process subject. In *2021 IEEE Frontiers in Education Conference (FIE)*. 1–7. https://doi.org/10.1109/FIE49875.2021.9637428

[16] E. Fernandes, J. Oliveira, and E. Figueiredo. 2016. Investigating how features of online learning support software process education. In *2016 IEEE Frontiers in Education Conference (FIE)*. 1–8. https://doi.org/10.1109/FIE.2016.7757681

[17] Thaís Ferreira, Davi Viana, Juliana Fernandes, and Rodrigo Santos. 2018. Identifying Emerging Topics and Difficulties in Software Engineering Education in Brazil. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering* (Sao Carlos, Brazil) *(SBES '18)*.

[18] A Fontão, R Santos, A Dias-Neto, et al. 2016. MSECO-DEV: Application development process in mobile software ecosystems. In *Proceedings of the international conference on software engineering and knowledge engineering (SEKE2016)*. 317–322.

[19] Erik M Fredericks, Byron DeVries, and Betty HC Cheng. 2014. Towards run-time adaptation of test cases for self-adaptive systems in the face of uncertainty. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 17–26.

[20] Laura García-Borgoñon, Miguel A Barcelona, Julián Alberto García-García, M Alba, and María José Escalona. 2014. Software process modeling languages: A systematic literature review. *Information and Software Technology* 56, 2 (2014), 103–116.

[21] Görkem Giray. 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software* 180 (2021), 111031. https://doi.org/10.1016/j.jss.2021.111031

[22] Ronald Jabangwe, Henry Edison, and Anh Nguyen Duc. 2018. Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software* 145 (2018), 98–111.

[23] Jørn Johansen, Ricardo Colomo-Palacios, and Rory V O'Connor. 2016. Towards a manifesto for software process education, training and professionalism. In *International Conference on Software Process Improvement and Capability Determination*. Springer, 98–105.

[24] André Koscianski and Michel dos Santos Soares. 2007. *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec Editora.

[25] Nane Kratzke and Peter-Christian Quint. 2017. Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software* 126 (2017), 1–16.

[26] Marco Kuhrmann, Daniel Méndez Fernández, and Jürgen Münch. 2013. Teaching Software Process Modeling. In *Proceedings of the 2013 International Conference on Software Engineering* (San Francisco, CA, USA) *(ICSE '13)*. IEEE Press, 1138–1147.

[27] Xabier Larrucea, Annie Combelles, John Favaro, and Kunal Taneja. 2017. Software engineering for the internet of things. *IEEE Software* 34, 1 (2017), 24–28.

[28] Hanyan Liu, Samuel Eksmo, Johan Risberg, and Regina Hebig. 2020. *Emerging and Changing Tasks in the Development Process for Machine Learning Systems*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/3379177.3388905

[29] Hanyan Liu, Samuel Eksmo, Johan Risberg, and Regina Hebig. 2020. Emerging and Changing Tasks in the Development Process for Machine Learning Systems. In *Proceedings of the International Conference on Software and System Processes* (Seoul, Republic of Korea) *(ICSSP '20)*. Association for Computing Machinery, New York, NY, USA, 125–134. https://doi.org/10.1145/3379177.3388905

[30] Victor Moura and Gleison Santos. 2018. ProcSoft: A Board Game to Teach Software Processes Based on ISO/IEC 29110 Standard. In *Proceedings of the 17th Brazilian Symposium on Software Quality* (Curitiba, Brazil) *(SBQS)*. Association for Computing Machinery, New York, NY, USA, 363–372.

[31] OMG. 2008. *Software & Systems Process Engineering Meta-Model Specification - Version 2.0*. Technical Report. Object Management Group (OMG). https://www.omg.org/spec/SPEM/2.0/PDF.

[32] Ann Osborne O'Hagan, Gerry Coleman, and Rory V. O'Connor. 2014. Software Development Processes for Games: A Systematic Literature Review. In *Systems, Software and Services Process Improvement*, Béatrix Barafort, Rory V. O'Connor, Alexander Poth, and Richard Messnarz (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 182–193.

[33] Pankesh Patel and Damien Cassou. 2015. Enabling high-level application development for the Internet of Things. *Journal of Systems and Software* 103 (2015), 62–84.

[34] Ian Sommerville. 2011. *Engenharia de software*. Pearson Brasil.

[35] S. Tiwari and S. Singh Rathore. 2019. Teaching Software Process Models to Software Engineering Students: An Exploratory Study. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*. 308–315. https://doi.org/10.1109/APSEC48747.2019.00049

[36] Murat Yilmaz, Serdar Tasel, Eray Tuzun, Ulas Gulec, Rory V. O'Connor, and Paul M. Clarke. 2019. Applying Blockchain to Improve the Integrity of the Software Development Process. In *Systems, Software and Services Process Improvement*, Alastair Walker, Rory V. O'Connor, and Richard Messnarz (Eds.). Springer International Publishing, Cham, 260–271.

[37] M. Younas, D. N. A. Jawawi, A. K. Mahmood, M. N. Ahmad, M. U. Sarwar, and M. Y. Idris. 2020. Agile Software Development Using Cloud Computing: A Case Study. *IEEE Access* 8 (2020), 4475–4484. https://doi.org/10.1109/ACCESS.2019.2962257