Journal of Software Engineering Research and Development, 2023, 11:14, doi: 10.5753/jserd.2023.3186

Modeling software processes from different domains using SPEM and BPMN notations: An experience report of teaching software processes

Carla Bezerra [**Federal University of Ceará** | ***carlailane@ufc.br***]
Emanuel Coutinho [**Federal University of Ceará** | ***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 the understanding for 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 teaching 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. In the discipline of software processes, we applied a practice for defining and modeling processes in various application domains, such as: IoT, cloud, mobile, critical systems, self-adaptive systems, machine learning, blockchain and games. The processes were modeled in the Software Systems Process Engineering Metamodel (SPEM) and Business Process Model and Notation (BPMN) notations based on references from the literature for each domain. We evaluated the process modeling practice with the SPEM and BPMN in two classes of the software processes discipline, and we had discussions about the use of the two notations applied to the different domains. In general, students reported good experiences in defining processes, highlighting the importance of practical modeling applications for professional life. As the main results of the study in teaching process modeling, we have that: (i) students accepted HEFLO tool better than EPF Composer tool; (ii) most students are not aware of specific domains and that anticipating the study of these domains in the discipline is a good strategy; and, (iii) the students also highlighted the need for more support for the two notation tools.

Keywords: *Software Process, Systems Domain, Education*

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 (Aleem et al., 2016). 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 (Calderón et al., 2018).

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 (Patel and Cassou, 2015), machine learning (Giray, 2021), blockchain (Yilmaz et al., 2019), Internet of Things (IoT) (Motta et al., 2023), startups (Pizzini et al., 2021) and self-adaptive systems (Andersson et al., 2013). However, each domain has activities specific to its development process, often requiring a different process than traditional software development.

A software process can be seen as the set of activities, methods, practices and transformations that guide people in the production of software (Moura and Santos, 2018). The proper definition of software processes requires understanding the particularities of the teams and organizations that will use them and specialized technical knowledge in Software Engineering (Moura and Santos, 2018). 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 (Johansen et al., 2016). Thus, software development teams use, formally or informally, a process to perform the tasks that will culminate in the final software product (Moura and Santos, 2018).

Most university curricula consider software processes to be on the fringes of Software Engineering (SE) (Kuhrmann et al., 2013). Since building software processes from scratch involves considerable effort, process tailoring should be practiced to adapt existing processes deriving new alternate ones that fit specific needs (Pillat and Oliveira, 2016). Courses related to systems modeling, such as Requirements, Systems Design Analysis, and Software Engineering, use diagrams to represent information (Pinheiro et al., 2022). And many times, sequences of activities, their relationships and generated products need to be specified.

Software process models are a way to keep organizations, projects and people together (Kuhrmann et al., 2013). 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 (Kuhrmann et al., 2013). 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 better to understand systems or environments (Coutinho et al., 2017). Also, models are important because they allow the representation of ideas, allowing analysis and comparison (Alencar et al., 2020). 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 (Chaves et al., 2015). Both in academia and industry, SPEM (Software & Systems Process Engineering Metamodel) and BPMN (Business Process Model and Notation) notations are the most used for modeling software processes (Castellanos Ardila et al., 2022). SPEM notation has been widely used by several studies in the academy because it has a language standardized by the Object Management Group (OMG) (de la Vara et al., 2020; Pazin et al., 2022), in addition to defining a generic framework for process modeling. EPF Composer tool supports SPEM notation facilitating process modeling. As for the BPMN notation, it is also the OMG standard for specifying process models and it has been adopted frequently as the industry standard (Moyón et al., 2020; Marin et al., 2023). Furthermore, the BPMN notation is supported by several tools. Teaching software process modeling is challenging, mainly because it emphasizes theory and offers few practical exercises (Chaves et al., 2015).

Software process education is an important software engineering field requiring a more practical and realistic approach to learning and teaching (Calderón et al., 2018; De Sena Quaresma and Oliveira, 2021). Several studies reported their experiences in learning and teaching software process models and described recommendations and encountered challenges (Kuhrmann et al., 2013; Calderón et al., 2018; Calderón et al., 2018, 2019; De Sena Quaresma and Oliveira, 2021).

Considering the described context, professors 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 adopted methodologies today are based on expository classes with little efficiency (Tiwari and Singh Rathore, 2019). Also, the modeling of software processes is often complicated to be applied in the disciplines in an applied manner.

In our previous work Bezerra and Coutinho (2022), we presented an experience report on the application of process modeling with SPEM for different domains in two classes of software processes. In the current work, we present other experience report teaching the definition and modeling of software processes in different domains, by extending the previous study Bezerra and Coutinho (2022) by changing the process modeling methodology to use BPMN for new process domains. Thus, this work consists of presenting two reports of experience in the classroom.

We applied the new methodology in a software processes class with 46 students. We assessed students in modeling with SPEM and BPMN quantitatively and qualitatively. In general, students reported good experiences in defining processes, highlighting the importance of practical modeling applications for professional life. As the main results of the study in teaching process modeling, we have that: (i) students accepted HEFLO tool better than EPF Composer tool; (ii) most students are not aware of specific domains and that anticipating the study of these domains in the discipline is a good strategy; and, (iii) the students also highlighted the need for more support for the two notation tools.

The rest of the paper is structured as follows. In Section 2, the fundamental concepts used in this work are presented. Section 3 presents the related work. Section 4 presents the work methodology for SPEM notation, with planning, execution and results discussions. Section 5 presents the work methodology for BPMN notation, with planning, execution and results discussions. In Section 6, some discussions are presented and the work's limitations are described. Finally, Section 7 shows the conclusions and future work.

2 Background

This section describes some theoretical aspects of SPEM and BPMN notations, used in this work.

2.1 SPEM Notation

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 (García-Borgoñon et al., 2014). 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 (OMG, 2008).

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 method content and the process. On the right side of the diagram, 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 (OMG, 2008).

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

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

tool for teaching software process modeling, based on several models and frameworks to support the definition of processes in several application domains.

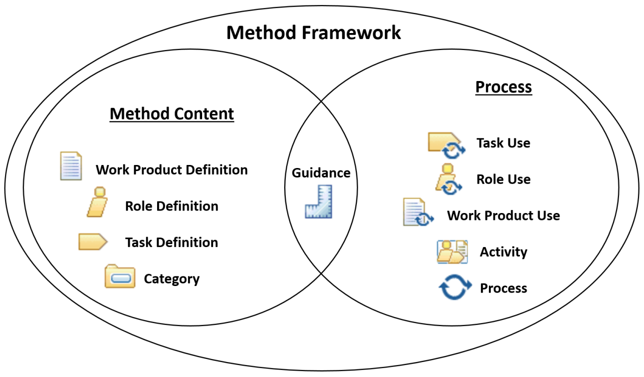

Figure 1. Key terminology defined in this specification mapped to Method Content versus Process (based on OMG (2008))

2.2 BPMN Notation

Business Process Model and Notation (BPMN) is a standard for business process modeling that provides graphical notation for specifying business processes maintained by the OMG (Omg et al., 2011). The objective of BPMN is to support business process modeling for both technical users and business users, by providing notation that is intuitive to business users, yet able to represent complex process semantics (von Rosing et al., 2015).

BPMN is constrained to support only the concepts of modeling that are applicable to Business Processes. This means that other types of modeling done by organizations for business purposes are out of scope for BPMN. Therefore, the following are aspects that are out of the scope of this International Standard: (i) Definition of organizational models and resources, (ii) Modeling of functional breakdowns, (iii) Data and information models, (iv) Modeling of strategy and (v) Business rules models (Omg et al., 2011).

The five basic categories of BPMN elements are: (1) Flow Objects, (2) Data, (3) Connecting Objects, (4) Swimlanes and (5) Artifacts. (1) Flow Objects are the main graphical elements to define the behavior of a Business Process. There are three Flow Objects: events, activities and gateways. (2) Data is represented with the four elements: data objects, data inputs, data outputs and data stores. There are four ways of connecting the Flow Objects to each other or other information. There are four Connecting Objects (3): sequence flows, message flows, associations and data associations. There are two ways of grouping the primary modeling elements through (4) Swimlanes: pools and lanes. (5) Artifacts are used to provide additional information about the Process. There are two standardized Artifacts, but modelers or modeling tools are free to add as many Artifacts as necessary. There could be additional BPMN efforts to standardize a larger set of Artifacts for general use or for vertical markets. The current set of Artifacts includes: group and text annotation (Omg et al., 2011). Figure 2.2 presents some elements of BPMN notation, in version 2.0.

Currently, there are several tools for modeling processes in BPMN (Medoh and Telukdarie, 2017). We chose the

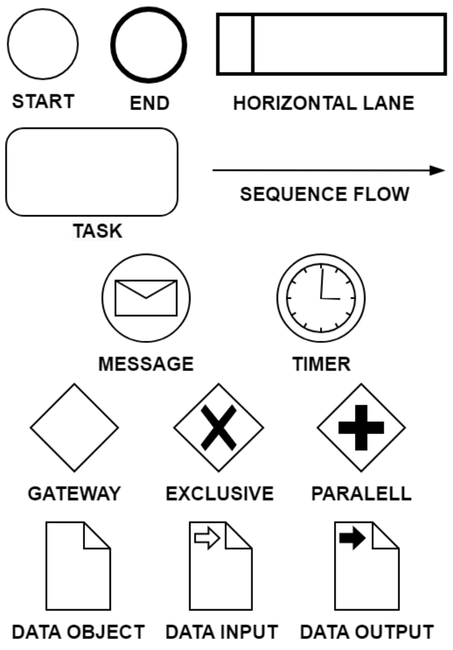

Figure 2. Examples of some BPMN notation elements, in version 2.0.

HEFLO[2] tool to use in our experience report on teaching BPMN modeling. The tool has a Portuguese version and a free academic license. In addition, the tool is web and allows the construction of processes collaboratively. It is also possible to download the entire process in a pdf file generated by the tool itself and detail the steps of each activity, in addition to the attachment of artifacts.

3 Related Work

This section describes some works related to this research. A brief discussion on Software Domain Processes is conducted, followed by reports on Teaching Software Processes with SPEM and BPMN.

3.1 Software Domain Processes

The development environment and method play a vital role in the success of software development (Younas et al., 2020). 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 (Osborne O'Hagan et al., 2014; Aleem et al., 2016), Internet of Things (Younas et al., 2020), mobile applications (Fontão et al., 2016; Jabangwe et al., 2018), self-adaptive systems (Andersson et al., 2013), machine learning systems (Liu et al., 2020), 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 (Andersson et al., 2013), as they are often critical systems that need to operate continuously. In traditional change

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

processes, changes are implemented during scheduled downtime and, as a consequence, continuous operation is not possible.

Osborne O'Hagan et al. (2014) 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 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. (2016) 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. (2020) 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. (2016) 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. (2013) 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, they argued 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. (2020) 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.

We emphasize that different domains have specific needs. Regardless of using a specific notation for defining software development processes, it is important to emphasize that each domain needs to be understood so that the modeling is adequate and useful for users. The previously mentioned articles served to identify some domains, such as games Osborne O'Hagan et al. (2014); Aleem et al. (2016), cloud computing Younas et al. (2020), mobile applications Fontão et al. (2016), self-adaptive systems Andersson et al. (2013), and machine learning Liu et al. (2020). Some of these articles were used in the course as a basis for knowledge in the domain to be studied, helping students in terms of knowledge.

3.2 Teaching Software Processes with SPEM

The teaching of software processes using SPEM has been discussed in some works identified in the literature in recent years (Kuhrmann et al., 2013; Fernandes et al., 2016; Calderón et al., 2018; Chaves et al., 2015; Moura and Santos, 2018).

Kuhrmann et al. (2013) 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. (2016) 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. (2018) 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. (2015) 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 (2018) 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 (Fernandes et al., 2016; Calderón et al., 2018; Moura and Santos, 2018) 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.

3.3 Teaching Software Processes with BPMN

The teaching of software processes with BPMN has been discussed in some works identified in the literature in recent years (Pillat and Oliveira, 2016; Hasić et al., 2020; Enríquez et al., 2019).

Pillat and Oliveira (2016) presented a new representation structure for software process tailoring that is based on high-level operations defined as BPMN extension concepts and associated support mechanisms. Such structure intents to contribute allowing the specification, record, and traceability of detailed control-flow variations required in the context of software process tailoring as well as facilitating the specification of such variations and improving their understanding.

Hasić et al. (2020) studied the modeling of IoT processes by comparing the standard BPMN approach and the combination of BPMN and Decision Model and Notation (DMN). Three cases with increasing need for context aggregation are modeled according to both techniques, leading to an analysis of the capability of the approaches to support IoT processes in terms of high-level context-awareness, scalability and complexity, flexibility, and decision logic reusability. They demonstrated that in cases where a need for complex context aggregation decision logic is present, the combination of BPMN and DMN provides the required support, even for the complex cases, and performs better than BPMN on its own.

Enríquez et al. (2019) performed an empirical analysis to evaluate the advantages of applying BPM in the implementation of innovative and dynamic teaching activities. Using this methodology, they designed RubricaSoft, a BPM system focused on providing dynamic educational processes. It automates multiple tasks, including peer evaluation, information integration and the management of deadlines. The results have been very promising from the viewpoint of the three axes upon which the evaluation has been carried out: satisfaction of students, improvement in academic results and increase in the productivity of teachers. In one of the processes, the time spent by the teacher has been reduced by 80% and student participation increased by 41%.

All these works reported experiences with the use of BPMN for teaching software processes, as well as our proposal. Pillat and Oliveira (2016) reinforced the use of notation for process specification, Hasić et al. (2020) applied it to the IoT domain, and Enríquez et al. (2019) highlighted productivity and reduction of time spent by the teacher on activities. In our proposal we want to present the experiences of using BPMN for the elaboration of software development processes, and analyze the benefits of its use.

4 Modeling different domains with SPEM

This section presents an experience report on modeling processes from different domains using the SPEM notation with the EPF Composer tool. We present the methodology that was applied in two classes of software processes. The chosen domains for process modeling were: self-adaptive systems, mobile, cloud, IoT, games and critical systems. The processes were elaborated using the specificities of each domain and also based on agile methodologies selected by the teams, and later the processes were modeled in EPF Composer. In the end, we evaluated the proposed methodology by analyzing the modeling of the domains.

4.1 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 3 illustrates the three steps and the sequence of activities. The details of each step are described below.

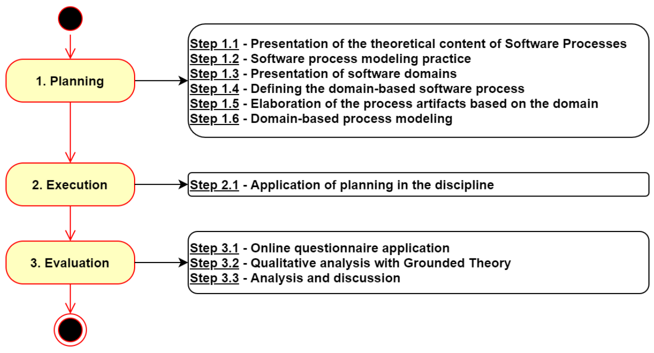
1. Planning
Step 1.1 - Presentation of the theoretical content of Software Processes
Step 1.2 - Software process modeling practice
Step 1.3 - Presentation of software domains
Step 1.4 - Defining the domain-based software process
Step 1.5 - Elaboration of the process artifacts based on the domain
Step 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 3. Representation of the work methodology

In Figure 3, 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; and (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 (Corbin and Strauss, 2014) methodology, inspired by the analysis approach presented in Ferreira et al. (2018). 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 (Corbin and Strauss, 2014). 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 (Corbin and Strauss, 2014). 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.

4.2 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 (Sommerville, 2011) and the Software Quality book of Koscianski (Koscianski and dos Santos Soares, 2007). 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 (OMG, 2008), 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.

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. Table 3 structure is used for definition. In the structure of Table 3, a process comprises two or more subprocesses. Each sub-process has a set of related activities. The defined process must contain the subprocesses of Requirements, Analysis and Design, Implementation, Tests and Project Manage-

Table 2. Theoretical reference for domains

Domain	Theoretical Reference
Self-Adaptive Systems	(Andersson et al., 2013; De Lemos et al., 2013)
Mobile Applications	(Fontão et al., 2016; Jabangwe et al., 2018)
Cloud Applications	(Kratzke and Quint, 2017; Cito et al., 2015)
Internet of Things	(Patel and Cassou, 2015; Larrucea et al., 2017)
Games Applications	(Osborne O'Hagan et al., 2014; Aleem et al., 2016)
Critical Systems	(Abdelaziz et al., 2015; Carrozza et al., 2018)

ment.

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.

4.2.1 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 it 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 or Keep All Objectives Satisfied) to specify the objectives of the system, requirements and uncertainties (Cheng et al., 2009). 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 Fredericks et al. (2014). 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[3].

4.3 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.

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

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%).

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. Most teams spent approximately 16 hours. 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 4 shows the methodologies, process models and frameworks used by students in constructing their processes. The student could select more than one option. The highlight was for Scrum, the most mentioned, but it was not used in all processes.

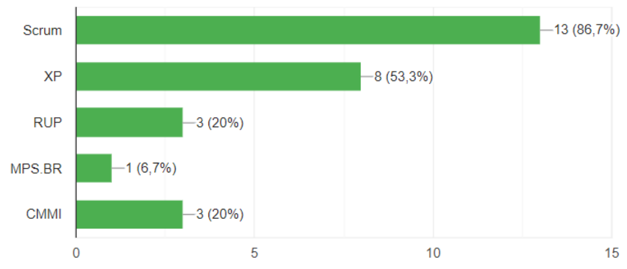

Figure 4. Methodologies, process models and frameworks used to build processes

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) (Corbin and Strauss, 2014) methodology using the Atlas.ti tool[4]. 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 (Corbin and Strauss, 2014). 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 (Corbin and Strauss, 2014). 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.

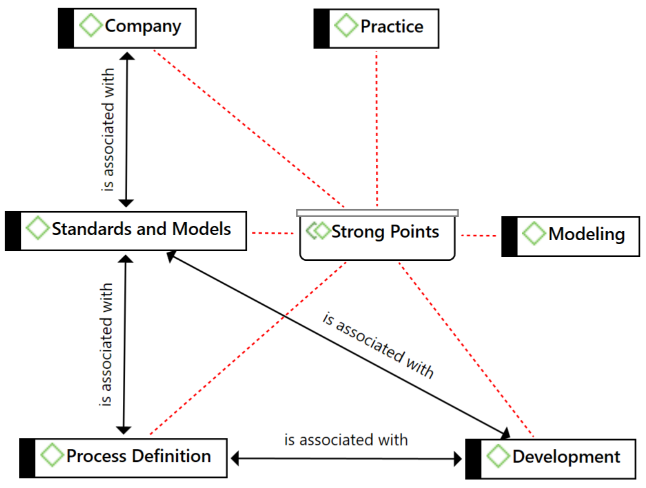

Figure 5. Strong points

Figure 5 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 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, re-

[4]`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

ported by P15 in "*Helps to better understand the structure of a process and its importance in development*".

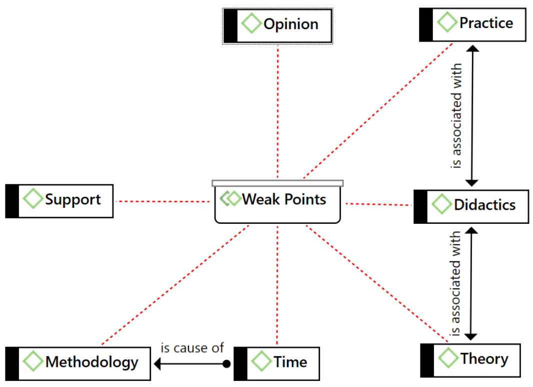

Figure 6. Weak points

Figure 6 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 7 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*

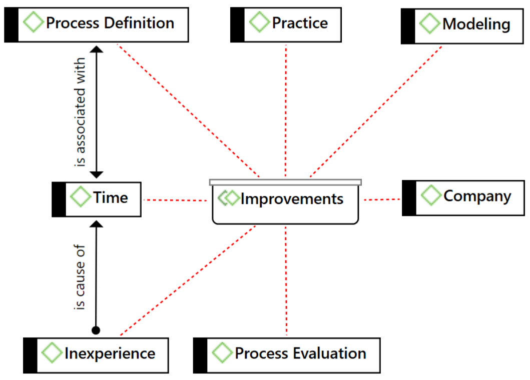

Figure 7. Improvements

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*".

4.4 Discussion

Figure 4 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 they 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.

5 Modeling different domains with BPMN

Based on our previous experience modeling processes with SPEM notation, we had several reports of difficulties faced in modeling, mainly related to using the EPF Composer tool. Another difficulty faced was the students' lack of maturity with the different software domains. In this way, we changed the discipline's final work of software processes to explore the use of BPMN notation, which is widely used in industry. Moreover, we also changed the methodology for the early study of new domains, and introduce other domains to students.

This section presents the second experience report for modeling processes from different software domains using the BPMN notation. In addition to using the BPMN notation, we changed some parts of the teaching methodology and we used new domains, such as: machine learning, blockchain and big data. We used a larger class of software processes, with different software processes, to apply the new methodology and evaluate the new teaching approach. We also contemplated further questions for evaluating the teaching methodology and process modeling.

5.1 Methodology

For modeling domain processes with BPMN, we changed the methodology presented in Section 4.1 of the previous report. Some difficulties identified in the previous application with the software process groups were related to the process modeling tool and the difficulty understanding the domain. In the same way as the previous methodology, the methodology was divided into planning, execution and analysis (see Figure 3). We changed only the planning part of the discipline, anticipating the investigation and presentation of the selected domains for modeling the processes. The discipline planning steps were: (1) presentation of the theoretical content of processes; (2) investigation and presentation of concepts about domains; (3) software process modeling practice; (4) defining the domain-based software process; (5) elaboration of process artifacts based on the domain; and (6) domain-based process modeling. There are also some changes in the execution of these steps, as described in the next section.

5.2 Discipline Planning

Step 1 - Presentation of the theoretical content of Software Processes: The theoretical concepts presented in the discipline of software processes had some changes compared to the planning presented in the previous report in Section 4.2. Initially, a new bibliography of the Book of Modern Software Engineering by Valente (2020) was added, considering the previous bibliography, the Software Quality book of Koscianski and dos Santos Soares (2007). Regarding the content shown in Table 1, the agile Shape Up methodology (Singer, 2019) was added, process modeling with BPMN was added, and the CMMI maturity model was removed.
Step 2 - Investigation and presentation of concepts about domains: We anticipated this step in relation to the planning of the previous discipline. Initially, teams must select an application domain to investigate the concepts and characteristics of that domain. We also added new more current domains to Table 2 to include more students in the class where the methodology will be implemented. Table 6 illustrates the new domains and references. After investigating the characteristics of the domains, students must present the main characteristics and activities that could be incorporated into the process. In this way, students can expand their knowledge of the selected domain.

Table 6. Theoretical reference for new domains

Domain	Theoretical Reference
Embedded Systems	(Oshana and Kraeling, 2019; Üstünel, 2020)
Blockchain Applications	(Chakraborty et al., 2018; Vacca et al., 2021)
Machine Learning Applications	(Amershi et al., 2019; Nascimento et al., 2019)
Big Data Applications	(Laigner et al., 2018; Biesialska et al., 2021)
Multi-agent Systems	(Mascardi et al., 2019; Weyns et al., 2019)
Startups	(Kemell et al., 2020a,b)

Step 3 - Software process modeling practice: In the previous application of the methodology, we felt the students' difficulty with the EPF Composer tool. In this way, we changed the methodology to contemplate the BPMN notation that is well-used in the software industry for process modeling. We used the academic version of HEFLO tool[5] to teach BPMN notation. HEFLO tool was chosen because it is online and collaborative, incorporating details of the process with the steps of the activities and attaching the input and output artifacts of the activities. In this step, we taught the concepts of BPMN notation and perform a practice for teaching the HEFLO tool.

Step 4 - Defining the domain-based software process: This step was equivalent to the one presented in Section 4.2 for defining the process. The same structure shown in Table 3 was used to define the entire domain development process with at least the following subprocesses: Requirements, Analysis and Design, Implementation, Tests and Project Management.

Step 5 - Elaboration of the process artifacts based on the domain: This step is similar to the one described in Section 4.2. Students will identify the main artifacts generated by the activities of the process defined in the previous step. A template for standardizing artifacts is also provided. In the same way as the process activities, the artifacts must also adapt to the characteristics of each domain.

Step 6 - Domain-based process modeling: In this step, students will model the process defined in the previous step in the HEFLO tool in BPMN notation. The tool is online and collaborative, facilitating students to work as a team. It also allows detailing the steps of the macro activities and attaching the artifacts produced in the previous step. 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 the artifacts. As a delivery, the final published process is expected and customized according to the domain.

5.2.1 Execution

The teaching methodology was applied to the Software Engineering course's Software Process discipline in the second semester of 2021. The class had 46 students, most of them from the third semester. As in the previous experience report, the definition and modeling of the domain-specific software process correspond to the discipline's final work. All methodology presented in the previous section was applied remotely due to the COVID-19 pandemic. However, this did not affect the transfer of the discipline's concepts and tools. The work was carried out in teams of up to five students. The team with more affinity with the theme applied to define the respective process. A division draw was carried out when more than one team showed interest in the same domain.

In parallel with the presentation of the discipline's concepts, described in Table 1, the students began studying the concepts of the domains. This step was anticipated since many students did not have experience with the selected software domains, as mentioned in the report of previous disciplines. The teams had four weeks to study the domains and prepare a presentation with the domain's main characteristics and with which software development steps should be changed to adapt the process to the specific domain. This part consists of the first delivery of the final work of the processes discipline. The fact that the study of domains was anticipated for the beginning of the discipline was very positive for the teams, who could better understand the possible adaptations of the software development process for each domain. The selected domains are described in Tables 2 and 6, excluding the domains of Self-Adaptive Systems and Critical Systems domains.

The second delivery consisted of the initial definition of the process based on the template presented in Table 3. At this stage, the students based themselves on the characteristics of the domain and on the agile scrum and XP methodologies to define the macro activities of the process. For this delivery, it was estimated to take two weeks. Students received high-level process feedback to proceed with detailing the process for domain. As in the previous report, it was also noticed that the students defined many activities and had few adaptations for the domain. It was also noticed that there needed to be more adaptations to incorporate activities related to agile scrum and XP methodologies.

In the third delivery, the students delivered the detailed process with the steps defined for each activity and the artifacts. Corrections from the previous delivery were also delivered, further adapting the process to mastery and agile methodologies. For this delivery was expected three weeks. As with the previous delivery, feedback was also given on the detailed processes. In general, students should detail more the activities of the processes and the templates were not adherent to the activities.

The fourth delivery consisted of modeling the corrected process of the previous delivery in BMPN notation in the HEFLO tool. The execution of the delivery was foreseen in two weeks. A notation and tool training was previously carried out with the students. The tool's academic license enabled collaborative modeling with the teams and generated a pdf for the entire detailed process. It was also possible to detail the steps of each activity and attach the artifacts associated with the activities. A negative point of the tool is that it needed to generate the website of the process.

The last delivery consisted of the team's final presentation of the process reporting the weaknesses, strengths and lessons learned from the work. The presentation consisted of going through the activities and artifacts of the defined process and focusing on what was adapted for each domain. The developed processes are available at the link[6].

5.3 Evaluation

For the evaluation of the process modeling methodology for the domains applied in the discipline, an online questionnaire was answered by 34 students of the discipline Software Processes. Not all students in the class answered the questionnaire. However, the sample of 34 students includes the 10 processes defined by the teams. After a review of the subject's content and methodology, the questions previously ap-

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

[6]https://doi.org/10.5281/zenodo.7674965

plied underwent some changes in relation to the previous application. The questionnaire presented in Table 7 was composed of three types of questions: demographic (1 to 8), about software and tool processes (9 and 10), and opinion (11 to 15). We applied the questionnaire at the end of the course; therefore, not all students responded. However, we got answers from all 10 teams that developed processes. We made it very clear that student answers and processes would be used for academic and scientific purposes, with student agreement.

Table 7. Questionnaire applied to students

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

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, the following distribution occurred: 2 (5.9%) for 2017.1, 2 (5.9%) for 2019.1, 1 (2.9%) for 2019.2, 4 (11.8%) for 2020.1, 19 (55.9%) for 2021.1, and 6 (17.6%) for 2021.2. This information reveals the diversity of entry semesters of students who took the discipline.

Regarding the team, the distribution of students in the ten teams was as follows: Processes for Applications Based on Blockchain 5 (14.7%), Processes for Applications Based on Machine Learning 1 (2.9%), Processes for Big Data Applications 5 (14.7%), Processes for Applications for Multiagent Systems 2 (5.9%), Processes for Cloud Computing 5 (14.7%), Processes for Internet of Things (IoT) 4 (11.8%), Processes for Digital Games 4 (11.8%), Processes for Processes for startups 3 (8.8%), Processes for Safety Critical Systems 1 (2.9%), and Processes for Embedded Systems 4 (11.8%)

When asked about the student's professional experience in defining processes, the majority (28 - 82.4%) answered no, and only 6 (17.6%) answered yes.

The level of experience in the field of application of the students' team varied, focusing on intermediate, little and no experience: Intermediate experience 10 (29.4%), No experience 10 (29.4%), and Little experience 4 (41.2%).

The materials used by the teams to deepen the domain were diversified. There was a great mention of scientific articles, induced by the discipline. Also a lot of searches on websites, and technical blogs. Books and videos were also mentioned.

The time of dedication in the activities varied from 12h to more than 40h. However, most answered around 4h. The time divisions were: more than 20h - 2 (5.9%), 20h - 7 (20.6%), 18h - 2 (5.9%), 14h - 6 (17.6%), 12h - 6 (17.6%), 10h - 4 (11.8%), less than 10h - 5 (14.7%), and 2 (5.9%) were unable to determine.

Regarding the applicability of the process in the market, many students reported they are applicable: Applicable with few adaptations 15 (44.1%), Fully applicable 10 (29.4%), Applicable with many adaptations 8 (23.5%), and Not applicable 1 (2,9%).

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, HEFLO. Only Scrum (25) and XP (7) were explicitly mentioned by students in constructing their processes. Only one person claimed to have used both.

Using the same categories (Table 5), and proceeding with the qualitative analysis with GT, it was also possible to categorize all the same items. Care was taken to verify whether new categories would be added, but no need was perceived.

The 14 identified categories were, listed in order of decreasing frequency: Support (70), Theory (36), Process Definition (29), Modeling (23), Company (16), Methodology (15), Opinion (12), Development (13), Didactic (11), Inexperience (11), Practice (11), Time (6), Process Evaluation (4) and Standards and Models (3).

Additionally, 12 relationships between codes were identified: Company is associated with Process Definition, Development is associated with Process Definition, Didactics is associated with Methodology, Modeling is associated with Support, Inexperience is associated with Process Definition, Inexperience is associated with Support, Opinion is associated with Inexperience, Opinion is associated with Practice, Opinion is associated with Time, Practice is associated with Theory, Process Definition is associated with Time, Support is associated with Theory.

Figure 8 shows the strengths identified in the research, and their codes and relationships. The relationship *Practice is associated with Theory* highlighted the importance seen by students in the relationship between theory and practice, as highlighted by P18 with "*The theory initially studied helped a lot to have an overview of processes, the practice helped to go deeper into processes and also the chance to know more about a specific domain*". The relationship *Company is associated with Process Definition* reports the importance of practical experiences in defining processes for professional life, highlighted by P34 in "*Learn more about agile methodologies that are widely used in the job market and about creating software processes*". The relationship *Didactics is associated with Methodology* reveals the recognition of the importance of the defined methodology so that the didactics is adequate for the students, as reported by P27 in "*Broad learning and well-defined structure*".

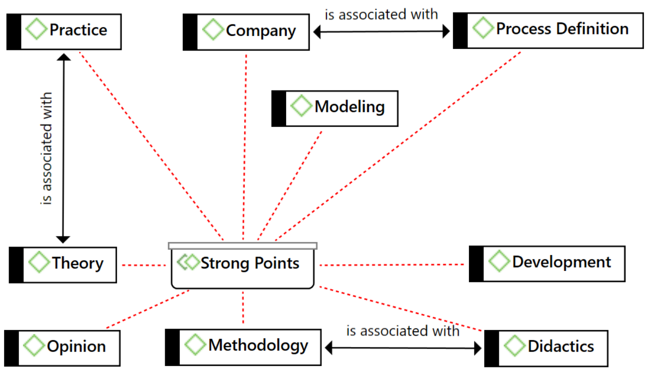

Figure 8. Strong points

Figure 9 shows the weaknesses identified in the research, with their codes and relationships. The relationship *Practice is associated with Theory* revealed that students considered a lot of theory in the discipline, and sometimes not compatible with practice, as in the comment of P13 in "*A load of mostly theoretical content, without much practical experimentation*". The relationship *Didactics is associated with Methodology* highlighted aspects that could be included in the methodology, and consequently adjusted for teaching in the classroom, mentioned by P12 in "*I think that referring to the course there was no weak point, but some things in the course were addressed, for example the works, I believe that some things could have been better elaborated... maybe if a class before, or in an asynchronous class, the class and the teacher... that would be much more interesting, because different themes could be addressed that were not addressed, but just as interesting.*".

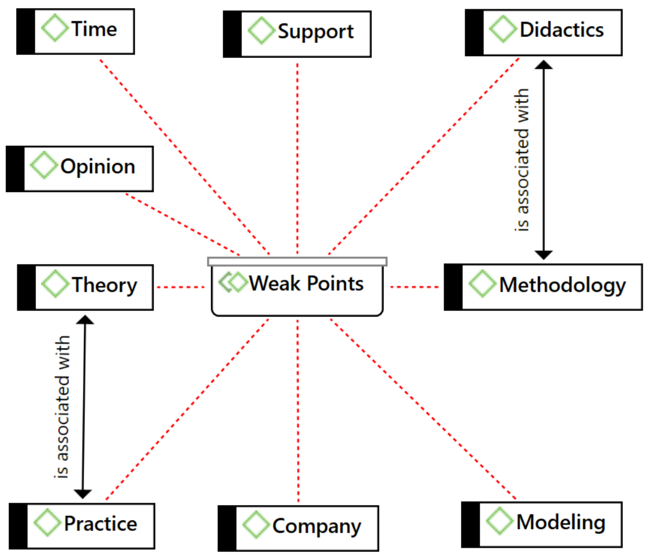

Figure 9. Weak points

Figure 10 shows the improvements identified in the research, with their codes and relationships. The relationship *Development is associated with Process Definition* showed the need for more details between the development and the definition of the process, and the addition of technical aspects when necessary, highlighted in P26's comments in "*Detail the process further of analysis and design*" and P29 with "*Could be improved by adding a bit more about the cloud architecture part and talking more in depth about development environments*".

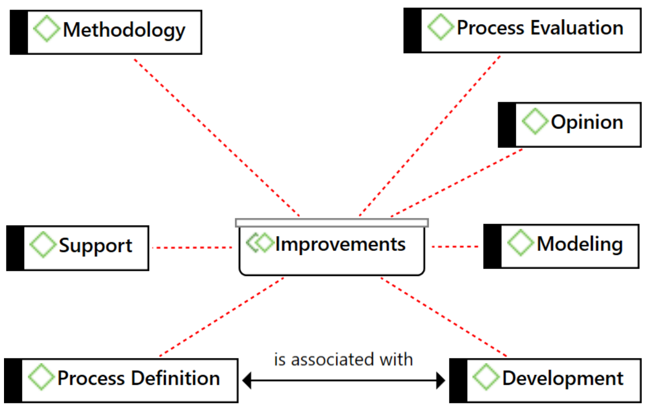

Figure 10. Improvements

Figure 11 shows the aspects related to the tool, identified in the research, with their codes and relationships. Relationships always had the *Support* category, denoting a need. The relationship *Modeling is associated with Support* showed some difference of opinion, with some contradictions. For example the comment of P4 with "*Easy to build the model*" and P31 "*Difficulties in collaboration when making the model*". The relationship *Inexperience is associated with Support* also showed disagreements among students, as in the comments of P14 with "*Despite not having previous experience with the tool, it was easy to adapt, since this tool uses the same shortcuts and mechanics as other applications, but sometimes I got disoriented with the simplicity of some features*" and P23 with "*The tool was adequate, but perhaps my team's inexperience with this specific tool may have hindered the process*".

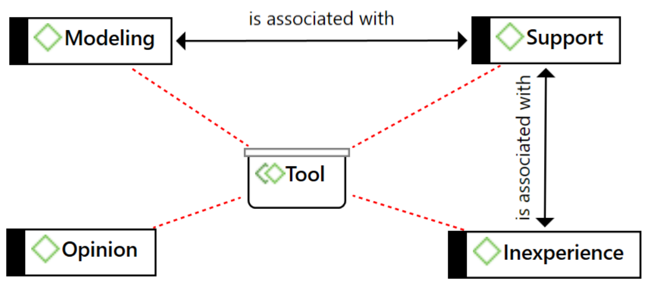

Figure 11. Tool

Figure 12 shows the difficulties identified in the research, with their codes and relationships. Most of the difficulties have some relation with the lack of support in the tool. The relationship *Inexperience is associated with Support* showed the relationship between the students' inexperience with the tool and the need for greater support, mentioned by P6 in "*The lack of real examples of well-detailed and documented processes that is accessible, making the initial visualization difficult, considering our inexperience in the area*". The relationship *Modeling is associated with Support* showed the same of the previous case, only related to modeling and support, highlighted by P14 in "*I believe that finding materials that would actually help the team to model the process, in ad-*

dition to my own inexperience in the area". Support is often related to a lack of documentation. The relationship *Support is associated with Theory* pointed out that students understand the importance of theory to better understand content and use the tool, mentioned by P22 in "*The content about these processes is very scarce*".

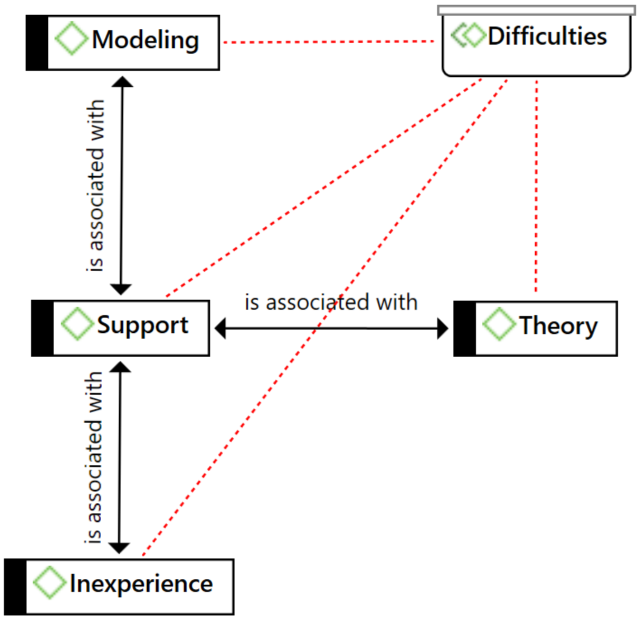

Figure 12. Difficulties

Figure 13 shows the students' perceptions identified in the research, with their codes and relationships. The students' perceptions were many in relation to learning, effort and complexity in carrying out the tasks of defining the process. The relationship *Opinion is associated with Practice* pointed out that, in general, the students had no experience in the practice of process modeling, mentioned by P2 in "*I think this is the first time using and putting it into practice, so it was very complicated, but then it was easier to understand*". The relationship *Opinion is associated with Inexperience* also indicated inexperience, mainly due to the complexity of the tasks, as highlighted by P6 with "*Depending on the domain this can be quite complex, especially considering the difficulties related to the inexperience of students in certain areas*". The relationship *Inexperience is associated with Process Definition* pointed out that, in general, students had no experience in defining processes, often due to a lack of theory, as highlighted by P14 in "*For not having any in-depth experience in the area of processes for the development of domains, many things were new to me, I believe that I did not have enough abstraction capacity to develop a process in all the molds that were given, so both me and the team had to break our heads in a few moments and research a lot to find materials that converged with our ideas*". The relationship *Process Definition is associated with Time* reinforced that the students felt a high load of effort for modeling the process, requiring dedication, reported by P27 with "*Defining a process requires a lot of attention and hours of dedication , plus you understand the basics of this domain, but it's very interesting and rewarding*". The relationship *Opinion is associated with Time* reinforced the time spent on process modeling activities, as well as the necessary dedication, highlighted by P8 in "*It is a complex activity that demands a lot of dedication and study*".

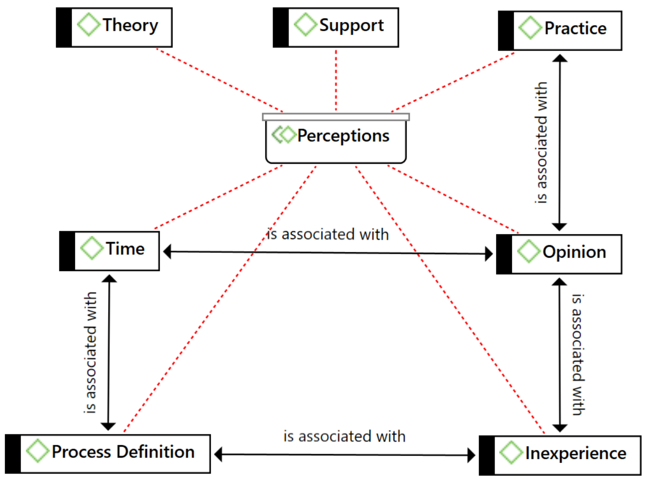

Figure 13. Students' perceptions

5.4 Discussion

In general, students reported good experiences in defining processes. A highlight was the observation about the importance of knowing the theory to be able to apply it in practice, and its importance for professional life.

The theory load could be less to explore more the practice. Despite the recognition of the need for theory, it was felt that there was too much. This is due to the inexperience of the teams in the selected domains and in learning the modeling tool, which was considered as theory for the students.

One of the questions investigated was whether the students already had professional experience in companies. Despite few "yes" answers (6 out of 34), this may have shown a certain maturity on the part of the students, mainly due to comments on the importance of modeling and defining processes in companies. However, in several responses it was noticed that even students who answered "no" also showed a certain degree of maturity.

Inexperience in software processes was pointed out in several responses. This inexperience is in the activity itself of defining and modeling the process, sometimes due to the lack of knowledge of the domain chosen by the team. It was also pointed out that this inexperience was in relation to the adopted tool and notation, several times also related to the lack of support.

Support in the tool was a perceived need both in the comments regarding the tool and in the students' perception of defining a process for a specific domain. The lack of support in the tool is a highlight because, despite being indicated by several students, the tool has a lot of documentation and support. There is a need to further investigate this lack of support, but an indication is that students did not dedicate themselves adequately to learning the tool and notation. This can also be an improvement for the next edition of the discipline.

Finally, the complexity and effort for modeling stood out in some responses. Complexity in defining the process and modeling was sometimes mentioned as a difficulty for students in the activities, along with the time or effort devoted to execution, reinforced by 9 of the 34 responses with 20 hours or more spent.

6 General Discussions

This section presents general discussions comparing the two reports. Additionally, some research limitations are presented.

6.1 Discussions between Reports

In addition to the difference in the modeling tool (SPEM and BPMN), some other differences between the applications can be highlighted. These differences were both in the methodology adopted in the discipline and in the results.

Regarding the discipline's methodology, in the second experience report we made the following changes: (i) anticipating the study of the main characteristics of each domain (Step 2) so that possible specific development activities for each domain could be identified; (ii) change of notation for process modeling to BPMN and consequently the tool; (iii) suggestions for new domains to be modeled such as blockchain, machine learning, big data and startups, and (iv) use only XP and SCRUM.

In the first report, students highlighted the importance of standards and models for professional life and application development. In the second report, the same feeling was also noticed, in addition to the need to know about the theory in order to better apply it in practice.

Regarding development maturity, for the first report this did not reflected much experience in the domain, with 12 responding out of 15 with little or no experience in the domain. In the second report, this same behavior also occurred, with an emphasis that the tool may have contributed to this inexperience.

Many defects were reported in the first report, in this case the EPF Composer tool, in addition to the lack or little documentation. With the change of tool in the second report (HEFLO), the need for more support was also noticed.

One difference was the addition of a question in the second report about the student's professional experience with software processes. Even with few answers (6 out of 34), it was understood that there was a certain maturity of these students in modeling and importance for companies.

In the responses of some students there was mention of the need for adaptations in the processes to be suitable for the domains and the market. There is also this need for adaptation so that the processes can also be used in the classroom, as each professor has their own methodology. This need was also identified in some works in the literature, such as Pillat and Oliveira (2016) and Moura and Santos (2018).

In both reports, students commented on a greater need for training and support. Fontão et al. (2016) also reinforced this aspect. As highlighted by Younas et al. (2020), there is a vast amount of tools for process and development support, and this requires training.

6.2 Threats to Validity and Research Limitations

One of the limitations of the work was only half of the class (15 students) answered the questionnaire, in the SPEM edition of the class. However, they were 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. In the BPMN edition of the class, que quantity of the awnwers was greater, with 34 answers.

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 the case of the HEFLO tool, several students pointed out problems with support and documentation. Again, a tool training can minimize these problems.

Another limitation is that the subject teacher only rated the quality process following the used methodologies. Ideally, people experienced in the domains would evaluate each process with the professor. 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.

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.

In general, the students did not have experience in the domains used to define the processes, which implied an extra effort in understanding the domains. A way to minimize this problem would be to have a training strategy in the domains, and not just indicate material and scientific articles.

The quality of the questionnaire's open responses may have influenced the conclusions, as some were superficial. Although the number of students who answered the questionnaire was greater in the second edition of the course, some responses were not of good quality, even with a reasonable amount of text. In addition, some questions may not have been well understood by the students, generating confusing answers or not having much relation with what was requested. In the continuity of the research (application with BPMN), the questionnaire underwent a revision, in order to try to minimize the misunderstanding of the questions.

Finally, another limitation was related to the acceptance of the notation, that is, how much better one notation was accepted by the students than the other. In fact, the notation was not evaluated, and the questionnaire discussed more the use of the tools. For this, a deeper analysis and comparison of students' experiences and learning of notations would be necessary. Therefore, we can not say with certainty that the use of one notation was better than the other, but we can say that there are indications that HEFLO was better accepted than EPF Composer.

7 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, in different classes. The contributions of this work were: (i) an approach to teaching software processes for different domains (self-adaptive systems, mobile applications, cloud applications, IoT systems, games applications, critical systems, embedded systems, blockchain applications, machine learning applications, big data applications, multi-agent systems and startups). (ii) discussions on the use of the EPF Composer and HEFLO tools in process modeling; and (iii) a qualitative analysis with Grounded Theory on the results of an online questionnaire.

As the main results of this study, we have that: (i) students accepted HEFLO tool better than EPF Composer tool; (ii) most students do not have knowledge in the specific domains used as a basis for process modeling; and, (iii) students highlighted the need for support for both modeling tools.

This work benefits students and professors 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) measure the performance of the applied methodology from the viewpoint of learning software development processes; (ii) use other tools for modeling BPMN notation; (iii) simulate a software process in a real environment; (iv) include peer review of processes by different teams in the methodology; (v) conduct an assessment of student learning using SPEM and BPMN and, (vi) compare which activities change in each domain in the software development process.

References

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

Aleem, S., Capretz, L. F., and Ahmed, F. (2016). Game development software engineering process life cycle: a systematic review. *Journal of Software Engineering Research and Development*, 4(1):6.

Alencar, I. R., Coutinho, E. F., Moreira, L. O., and Bezerra, C. I. M. (2020). A tool for software ecosystem models: An analysis on their implications in education. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, SBES '20, page 405–414, New York, NY, USA. Association for Computing Machinery.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300.

Andersson, J., Baresi, L., Bencomo, N., de Lemos, R., Gorla, A., Inverardi, P., and Vogel, T. (2013). *Software Engineering Processes for Self-Adaptive Systems*, pages 51–75. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bezerra, C. I. M. and Coutinho, E. F. (2022). Teaching software processes from different application domains. In Canedo, E. D., Viana, D., Garcia, V., Bezerra, C. I. M., de Sousa Santos, I., Gadelha, B., Machado, I., Soares, S., Kulesza, U., de França, B., Conte, T., Maldonado, J. C., Reinehr, S. S., Malucelli, A., Albuquerque, A. B., Santos, G., Barcellos, M. P., dos Santos, R. P., Lima, C., Monteiro, D., Damian, A., and Rocha, L., editors, *Proceedings of the XXI Brazilian Symposium on Software Quality, SBQS 2022, Curitiba, Brazil, November 7-10, 2022*, pages 29:1–29:10. ACM.

Biesialska, K., Franch, X., and Muntés-Mulero, V. (2021). Big data analytics in agile software development: A systematic mapping study. *Information and Software Technology*, 132:106448.

Calderón, A., Trinidad, M., Ruiz, M., and O'Connor, R. V. (2018). Teaching software processes and standards: a review of serious games approaches. In *International Conference on Software Process Improvement and Capability Determination*, pages 154–166. Springer.

Calderón, A., Trinidad, M., Ruiz, M., and O'Connor, R. V. (2019). An experience of use a serious game for teaching software process improvement. In *European Conference on Software Process Improvement*, pages 249–259. Springer.

Calderón, A., Ruiz, M., and O'Connor, R. V. (2018). A multivocal literature review on serious games for software process standards education. *Computer Standards & Interfaces*, 57:36 – 48.

Carrozza, G., Pietrantuono, R., and Russo, S. (2018). A software quality framework for large-scale mission-critical systems engineering. *Information and Software Technology*, 102:100–116.

Castellanos Ardila, J. P., Gallina, B., and Ul Muram, F. (2022). Compliance checking of software processes: A systematic literature review. *Journal of Software: Evolution and Process*, 34(5):e2440.

Chakraborty, P., Shahriyar, R., Iqbal, A., and Bosu, A. (2018). Understanding the software development practices of blockchain projects: A survey. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '18, New York, NY, USA. Association for Computing Machinery.

Chaves, R. O., von Wangenheim, C. G., Furtado, J. C. C., Oliveira, S. R. B., Santos, A., and Favero, E. L. (2015).

Experimental evaluation of a serious game for teaching software process modeling. *IEEE Transactions on Education*, 58(4):289–296.

Cheng, B. H., Sawyer, P., Bencomo, N., and Whittle, J. (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*, pages 468–483. Springer.

Cito, J., Leitner, P., Fritz, T., and Gall, H. C. (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*, pages 393–403.

Corbin, J. and Strauss, A. (2014). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 4 edition.

Coutinho, E. F., Viana, D., and dos Santos, R. P. (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)*, MISE '17, pages 47–53, Piscataway, NJ, USA. IEEE Press.

de la Vara, J. L., Marín, B., Ayora, C., and Giachetti, G. (2020). An empirical evaluation of the use of models to improve the understanding of safety compliance needs. *Information and Software Technology*, 126:106351.

De Lemos, R., Giese, H., Müller, H. A., Shaw, M., Andersson, J., Litoiu, M., Schmerl, B., Tamura, G., Villegas, N. M., Vogel, T., et al. (2013). Software engineering for self-adaptive systems: A second research roadmap. In *Software Engineering for Self-Adaptive Systems II*, pages 1–32. Springer.

De Sena Quaresma, J. A. and Oliveira, S. R. B. (2021). Teaching and learning strategies for software process subject. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–7.

Enríquez, F., Troyano, J. A., and Romero-Moreno, L. M. (2019). Using a business process management system to model dynamic teaching methods. *The Journal of Strategic Information Systems*, 28(3):275–291.

Fernandes, E., Oliveira, J., and Figueiredo, E. (2016). Investigating how features of online learning support software process education. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–8.

Ferreira, T., Viana, D., Fernandes, J., and Santos, R. (2018). Identifying emerging topics and difficulties in software engineering education in brazil. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, SBES '18.

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

Fredericks, E. M., DeVries, B., and Cheng, B. H. (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*, pages 17–26.

García-Borgoñon, L., Barcelona, M. A., García-García, J. A., Alba, M., and Escalona, M. J. (2014). Software process modeling languages: A systematic literature review. *Information and Software Technology*, 56(2):103–116.

Giray, G. (2021). A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software*, 180:111031.

Hasić, F., Serral, E., and Snoeck, M. (2020). Comparing bpmn to bpmn + dmn for iot process modelling: A case-based inquiry. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 53–60, New York, NY, USA. Association for Computing Machinery.

Jabangwe, R., Edison, H., and Duc, A. N. (2018). Software engineering process models for mobile app development: A systematic literature review. *Journal of Systems and Software*, 145:98–111.

Johansen, J., Colomo-Palacios, R., and O'Connor, R. V. (2016). Towards a manifesto for software process education, training and professionalism. In *International Conference on Software Process Improvement and Capability Determination*, pages 98–105. Springer.

Kemell, K.-K., Ravaska, V., Nguyen-Duc, A., and Abrahamsson, P. (2020a). Software startup practices–software development in startups through the lens of the essence theory of software engineering. In *Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21*, pages 402–418. Springer.

Kemell, K.-K., Risku, J., Strandjord, K. E., Nguyen-Duc, A., Wang, X., and Abrahamsson, P. (2020b). Internal software startups – a multiple case study on practices, methods, and success factors. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 326–333.

Koscianski, A. and dos Santos Soares, M. (2007). *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec Editora.

Kratzke, N. and Quint, P.-C. (2017). Understanding cloud-native applications after 10 years of cloud computing-a systematic mapping study. *Journal of Systems and Software*, 126:1–16.

Kuhrmann, M., Fernández, D. M., and Münch, J. (2013). Teaching software process modeling. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, page 1138–1147. IEEE Press.

Laigner, R., Kalinowski, M., Lifschitz, S., Salvador Monteiro, R., and de Oliveira, D. (2018). A systematic mapping of software engineering approaches to develop big data systems. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 446–453.

Larrucea, X., Combelles, A., Favaro, J., and Taneja, K. (2017). Software engineering for the internet of things. *IEEE Software*, 34(1):24–28.

Liu, H., Eksmo, S., Risberg, J., and Hebig, R. (2020). Emerging and changing tasks in the development process for machine learning systems. In *Proceedings of the Inter-*

national Conference on Software and System Processes, ICSSP '20, page 125–134, New York, NY, USA. Association for Computing Machinery.

Marin, J., Hurtado, J., Bastarrica, M., and Silvestre, L. (2023). Tailoring hybrid software processes in a medium-size software company. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, SAC '23, page 1042–1050, New York, NY, USA. Association for Computing Machinery.

Mascardi, V., Weyns, D., Ricci, A., Earle, C. B., Casals, A., Challenger, M., Chopra, A., Ciortea, A., Dennis, L. A., Díaz, Á. F., et al. (2019). Engineering multi-agent systems: State of affairs and the road ahead. *ACM SIGSOFT Software Engineering Notes*, 44(1):18–28.

Medoh, C. and Telukdarie, A. (2017). Business process modelling tool selection: A review. In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 524–528.

Motta, R. C., de Oliveira, K. M., and Travassos, G. H. (2023). An evidence-based roadmap for iot software systems engineering. *Journal of Systems and Software*, 201:111680.

Moura, V. and Santos, G. (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*, SBQS, page 363–372, New York, NY, USA. Association for Computing Machinery.

Moyón, F., Soares, R., Pinto-Albuquerque, M., Mendez, D., and Beckers, K. (2020). Integration of security standards in devops pipelines: An industry case study. In *Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21*, pages 434–452. Springer.

Nascimento, E. d. S., Ahmed, I., Oliveira, E., Palheta, M. P., Steinmacher, I., and Conte, T. (2019). Understanding development process of machine learning systems: Challenges and solutions. In *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–6.

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.

Omg, O., Parida, R., and Mahapatra, S. (2011). Business process model and notation (bpmn) version 2.0. *Object Management Group*, 1(4):18.

Osborne O'Hagan, A., Coleman, G., and O'Connor, R. V. (2014). Software development processes for games: A systematic literature review. In Barafort, B., O'Connor, R. V., Poth, A., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 182–193, Berlin, Heidelberg. Springer Berlin Heidelberg.

Oshana, R. and Kraeling, M. (2019). *Software engineering for embedded systems: Methods, practical techniques, and applications*. Newnes.

Patel, P. and Cassou, D. (2015). Enabling high-level application development for the internet of things. *Journal of Systems and Software*, 103:62–84.

Pazin, M., Dias, J., OliveiraJr, E., Aleixo, F. A., Kulesza, U., and Teixeira, E. N. (2022). Variability representation in software process with the smartyspem approach. In *UML-Based Software Product Line Engineering with SMarty*, pages 369–391. Springer.

Pillat, R. M. and Oliveira, T. C. (2016). A representation structure for software process tailoring based on bpmn high-level operations. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, page 1576–1579, New York, NY, USA. Association for Computing Machinery.

Pinheiro, F. V. S., Coutinho, E. F., Santos, I., and Bezerra, C. I. M. (2022). A tool for supporting the teaching and modeling of software ecosystems using ssn notation. *Journal on Interactive Systems*, 13(1):192–204.

Pizzini, A., Bortolo Vieira, R., Deda Gomes, R., Santos, G., Malucelli, A., and Reinehr, S. (2021). Software quality practices in growing startups: A qualitative study. In *XX Brazilian Symposium on Software Quality*, pages 1–10.

Singer, R. (2019). In *Shape Up: Stop Running in Circles and Ship Work that Matters*. 37signals LLC.

Sommerville, I. (2011). *Engenharia de software*. Pearson Brasil.

Tiwari, S. and Singh Rathore, S. (2019). Teaching software process models to software engineering students: An exploratory study. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)*, pages 308–315.

Üstünel, H. (2020). A project based innovative approach to an embedded systems course laboratory in software engineering education. *Computer applications in engineering education*, 28(1):160–166.

Vacca, A., Di Sorbo, A., Visaggio, C. A., and Canfora, G. (2021). A systematic literature review of blockchain and smart contract development: Techniques, tools, and open challenges. *Journal of Systems and Software*, 174:110891.

Valente, M. T. (2020). In *Engenharia de Software Moderna*. Independent publisher.

von Rosing, M., White, S., Cummins, F., and de Man, H. (2015). Business process model and notation—bpmn. In von Rosing, M., Scheer, A.-W., and von Scheel, H., editors, *The Complete Business Process Handbook*, pages 433–457. Morgan Kaufmann, Boston.

Weyns, D., Mascardi, V., and Ricci, A. (2019). Engineering multi-agent systems. *Lecture notes in computer science*, 11375.

Yilmaz, M., Tasel, S., Tuzun, E., Gulec, U., O'Connor, R. V., and Clarke, P. M. (2019). Applying blockchain to improve the integrity of the software development process. In Walker, A., O'Connor, R. V., and Messnarz, R., editors, *Systems, Software and Services Process Improvement*, pages 260–271, Cham. Springer International Publishing.

Younas, M., Jawawi, D. N. A., Mahmood, A. K., Ahmad, M. N., Sarwar, M. U., and Idris, M. Y. (2020). Agile software development using cloud computing: A case study. *IEEE Access*, 8:4475–4484.