


# A portal to catalog worked examples extracted from open source software projects to support the teaching of Software Engineering

Simone Tonhão  [ Universidade Estadual de Mato Grosso do Sul - UEMS | Universidade Estadual de Maringá - UEM | [siimone.franca@gmail.com](mailto:siimone.franca@gmail.com) ]

Thelma Colanzi  [ Universidade Estadual de Maringá - UEM | [teclopes@din.uem.br](mailto:teclopes@din.uem.br) ]

Igor Steinmacher  [ Northern Arizona University - NAU | [igor.steinmacher@nau.edu](mailto:igor.steinmacher@nau.edu) ]

## Abstract

Software Engineering is continually evolving, with new techniques, tools, and processes emerging to enhance software development. However, finding real-life examples that reflect this evolution can be challenging for instructors. Open Source Software (OSS) projects offer a valuable resource in this context, as they provide access to actual development projects and environments. Despite their potential, integrating these projects into the classroom involves several hurdles, including selecting suitable projects, preparing classes, and adapting to the open-source environment. This study aims to alleviate the challenges instructors face in adopting OSS projects for teaching Software Engineering. We developed an open portal to catalog worked examples from OSS projects, thereby supporting instructors in demonstrating real-world Software Engineering concepts and techniques. Utilizing Design Science Research, we followed the Relevance, Design, and Rigor cycles to construct this solution. The primary contribution of this work is the portal itself, which helps reduce the time instructors spend searching for relevant materials and resources. Additionally, we proposed a template to create, structure, and catalog these examples and developed guidelines to assist instructors in using the worked examples effectively. We conducted a series of studies with experienced Software Engineering instructors, which indicated that the portal could significantly mitigate the challenges associated with sourcing and updating real examples. The effectiveness of the examples was also assessed based on student perceptions, revealing that exposure to worked examples from OSS projects could engage students with real projects and challenges.

**Keywords:** *Software Engineering Education, Open Source Software Projects, Worked Examples*

## 1 Introduction

The beginning of your introduction is well-constructed, but I have made a few edits to enhance clarity, grammar, and style while maintaining the LaTeX commands and citations:

With the increasing demand for Information Technology professionals worldwide, the teaching of Software Engineering (SE) has encountered challenges in meeting market needs (Metrólho et al., 2022; Yamaguti et al., 2017). Software Engineering is constantly advancing with the creation of new techniques, tools, and processes aimed at improving software development. This dynamic landscape, where technologies rapidly become obsolete or mainstream, presents a challenging scenario for instructors who need to keep their teaching practices up-to-date (Montagner and Kurauchi, 2022; Pinto et al., 2017).

As a side effect of such rapid changes, Software Engineering instructors must regularly update their skills and lectures. Given their other responsibilities—such as hiring, researching, advising, and servicing (Hu et al., 2018; Holmes et al., 2018; Sadiku et al., 2012)—not all are willing or able to continuously update their teaching agendas. Worse, those willing to adapt often lack resources that can clearly demonstrate the applicability of specific Software Engineering concepts. Consequently, instructors frequently resort to using simplistic, toy examples in their classrooms. Although these examples might address timely practical issues, they often lack the maturity or breadth necessary for real software development

(Deng et al., 2020; Nascimento et al., 2018).

Open Source Software (OSS) projects provide an alternative for teaching Software Engineering. Engaging with OSS not only contributes to knowledge but also helps students align with professional software development and fosters creativity in planning and implementing project improvements Montagner and Kurauchi (2022); Pereira (2021). These projects allow students to practice design, development, and testing skills in real environments with actual stakeholders (Tan et al., 2021; Deng et al., 2020; Pereira and Pitxitxi, 2020). Furthermore, students gain experience in reading documentation and code, contributing to projects, adding new features, and identifying and correcting defects (Pereira, 2021; Pinto et al., 2017; Smith et al., 2014).

However, utilizing OSS projects in teaching can be challenging and requires significant effort to select appropriate projects for student engagement (Pereira and Díaz, 2022; Silva et al., 2020b; Pereira and Pitxitxi, 2020; Pinto et al., 2017; Morgan and Jensen, 2014; Smith et al., 2014). These projects must not be overly complex, as this could lead to comprehension difficulties, nor overly simplistic, as students may not see practical applications of the concepts and techniques (Pereira, 2021; Smith et al., 2014). Moreover, preparing lectures is another significant challenge (Silva et al., 2019; Pinto et al., 2017), often requiring many hours of work for the instructor (Buchta et al., 2006). According to Pinto et al. (2017), part of this difficulty arises from the need for instructors to familiarize themselves with OSS projects, as

information in some communities is not organized in a manner that facilitates understanding.

In response, this study aims to reduce the difficulties instructors face when adopting OSS projects in their Software Engineering courses by developing an open portal to catalog worked examples extracted from OSS projects. These examples support instructors in teaching Software Engineering concepts and techniques using real-world cases. Additionally, we have created guidelines to assist instructors in familiarizing themselves with and adopting *worked examples* (Chen et al., 2019).<sup>1</sup>

To achieve the proposed objectives, we conducted our research following the Design Science Research (DSR) paradigm, which focuses on constructing innovative artifacts to solve practical problems in specific domains (Hevner and Chatterjee, 2010). The DSR approach is iterative, composed of three interconnected cycles: Relevance Cycle, Design Cycle, and Rigor Cycle. The Relevance Cycle integrates the contextual aspects of the project with Design Science activities; the Design Cycle involves the construction, evaluation, and refinement of the artifact; and the Rigor Cycle ensures the project's innovation, relating to the foundations necessary for the development of the artifact and the contributions generated (Hevner, 2007).

In the *Relevance Cycle*, we conducted studies on the use of OSS projects in teaching Software Engineering to identify the difficulties faced by instructors in these educational environments. Additionally, we surveyed Software Engineering instructors to investigate the usage of examples and the challenges in finding suitable ones. Based on the studies carried out in the Relevance Cycle, we have defined the following requirements for building the portal:

- **Req1** – The portal should provide guidelines to instructors on how to create and standardize worked examples;
- **Req2** – The portal should openly provide worked examples to support the teaching of Software Engineering;
- **Req3** – The portal should guide instructors in applying worked examples when teaching Software Engineering;
- **Req4** – The portal must enable cataloging of worked examples that address different topics of Software Engineering.

In the *Design Cycles*, activities related to the development and evaluation of the cataloging portal were performed. In total, six cycles were executed, divided into two stages. The first stage involved three cycles focused on the creation and evaluation of the template for cataloging worked examples, which provides guidelines to guide the standardization, structuring, and cataloging of examples. The second stage included three cycles related to the use and evaluation of the portal prototype.

Finally, the *Rigor Cycle*, which focuses on the fundamentals and contributions of the project, ensures that the project is not merely routine and that it contributes effectively, ensuring innovation. The main foundations of this work are related to knowledge about Software Engineering education, OSS projects, and the use of worked examples.

<sup>1</sup> In the context of this work, a worked example is defined as an artifact composed of a problem, the steps to solve this problem, and a final result, and that problem was submitted to an OSS project (Chen et al., 2019).

The primary contribution of this research is the portal for cataloging worked examples extracted from OSS projects, which assists instructors in finding materials that truly exemplify the contents of Software Engineering. Additionally, we identified some of the difficulties instructors face when adopting OSS projects in teaching. We created guidelines to aid instructors in adopting worked examples for teaching Software Engineering. Evidence also suggests that the worked examples extracted from OSS projects can provide students with exposure to real projects and challenges. Furthermore, the portal can alleviate the challenges instructors face in searching for real examples, helping them stay up-to-date and proving to be a promising tool to support the teaching of Software Engineering.

The article is structured as follows: Section 2 provides background information on worked examples, and Section 3 presents the related work. Section 4 outlines the methodology adopted in a general manner. The relevance cycles of the research, as well as the design cycles, which have been divided into two stages, are discussed in Sections 5, 6, and 7. Section 8 addresses the rigor cycle applied to ensure the validity of the results. The limitations of the study are examined in Section 9, and the final conclusions are presented in Section 10, summarizing the main findings and outline possible directions for future research.

## 2 Background

According to Sweller et al. (1998), a worked example is an artifact focused on the statement of a problem, and the steps to solve that problem. As for Atkinson et al. (2003), a worked example consists in formulating a problem, in the solution steps, and in the final answer. Generally, students who receive worked examples tend to make fewer mistakes, solve similar problems more easily and quickly, and require less assistance from the instructor (Carroll, 1994). Skudder and Luxton-Reilly (2014), in their work, found evidence to suggest that the worked examples are an improvement over problem-solving approaches, both in terms of learning time and performance in similar problems.

This is because worked examples reduce the cognitive burden imposed by problem-solving, and facilitate the construction of cognitive schemes (Van Gog and Kester, 2012; Sweller et al., 1998). The principle of the worked examples shows that it is better to replace some of the practical problems with examples that demonstrate the solution of a given problem and enable students to first study these solutions before solving problems alone (Booth et al., 2015).

The worked example approach brings a number of benefits to learning, especially for students who do not have prior knowledge, and can be more efficient than learning through problem solving (Nivelstein et al., 2013; Van Gog and Kester, 2012). Based on the worked examples, it is possible to guide students to develop an understanding of principles and concepts, focusing on problematic states and solution steps, allowing students to build generalized solutions and schemes, decreasing cognitive effort (Schwonke et al., 2009; Sweller et al., 1998).

By learning through worked examples, students dedicate

more cognitive resources to the concepts that support correct solutions and make acquisitions of relevant schemes for knowledge. When students attempt to solve practical problems on their own they assume procedures to perform the resolution, these procedures may not be appropriate, in some cases inefficient and non-generalizable, and even incorrect (Sweller et al., 1998). When students learn through worked examples the main task is to understand the step-by-step of a solution (Wang et al., 2015). The worked examples with self-explanation consist of asking students to explain this step-by-step to themselves, reinforcing the understanding of the concepts and the ability to understand the steps described. Without self-explanation, the examples are stored in a textual and superficial way (Wang et al., 2015; Booth et al., 2015).

The use of worked examples has been widely explored in teaching various areas of knowledge. Studies such as those by Nievelstein et al. (2013) and Schwonke et al. (2009) show that worked examples can reduce cognitive load, making learning more efficient than an approach based solely on problem-solving. Furthermore, Van Gog et al. (2011) highlights that presenting similar problems after worked examples can motivate students, making the combination of worked examples and problem-solving more effective than the isolated use of worked examples. In the work of Rourke and Sweller (2009), the idea cited by Van Gog et al. (2011) was further evidenced, as the use of worked examples proved superior to problem-solving, significantly reducing the cognitive load required of students. Additionally, the level of students' prior knowledge is also a relevant factor: novice students benefit more from worked examples than advanced students. Studies suggest that worked examples are a valuable tool in teaching, reducing cognitive load and facilitating understanding, especially for less experienced students.

In the field of Computer Science, the use of Worked Examples has been more common in programming education, especially at the elementary and secondary levels. An example is the work of Bofferding et al. (2022), in which correct, incorrect, and incomplete worked examples were used to assist elementary school students in understanding programming concepts. These examples were incorporated into a block-based programming game, where each level focused on a different example. The results indicated that students with little programming experience benefited from the approach. Thus, the examples proved to be effective for students with limited prior knowledge, in line with the work of Nievelstein et al. (2013).

In the study by Toukiloglou and Xinogalos (2022), worked examples were also used in the context of games to teach programming at the elementary level. Like in Bofferding's work, the game also involved block programming, and the worked examples were presented before the challenges to demonstrate concepts to the students before they tackled problem-solving tasks. The results suggest an increase in the efficiency of student learning when worked examples in the game are combined with problem-solving.

In higher education, some studies have also utilized worked examples in the context of programming education. Garces et al. (2022) explored worked examples in teaching programming to Engineering Technology students. Two ap-

proaches were used in this study: debugging worked examples and code commenting. Students worked in two groups, with one group receiving an example containing code and the task being to comment on the code. The other group received an example with partially functional code and had to make it functional while also commenting on the code about the changes made. The study results suggest that worked examples helped students with no prior knowledge to be more successful in the subject.

In the work by Gaweda et al. (2020), the TYPOS platform was developed, focusing on practicing low-level skills through interactive worked examples, such as code typing exercises. The research involved Computer Science students, and the results revealed that students who consistently completed the code typing exercises achieved higher grades and experienced greater learning gains compared to those who practiced less frequently.

Although worked examples are popular in programming learning in the field of Computer Science, their application for SQL learning is limited, with only two studies found in the literature. The work of Chen et al. (2019) explored the use of worked examples in a database course. In this approach, the worked examples were merged with problem-solving, and with incorrect worked examples (they present incorrect solutions for a given problem) in an intelligent tutoring system. The results show that students who participated in the experiment improved their conceptual, procedural, and debugging knowledge. Furthermore, it can be observed that the use of worked examples followed by problem-solving can be effective in terms of learning, as concluded by the work in other areas presented above. It is important to emphasize that in this approach the incorrect worked examples proved to be more effective than the correct worked examples.

In the work by Akhuseyinoglu et al. (2022), worked examples were also employed in the Database course. For this purpose, a tool called DBQA was created to present interactive worked examples. The results showed that exploring more query execution steps in DBQA was associated with a higher success rate, fewer query construction attempts, and greater persistence during problem-solving.

In the field of Software Engineering, specifically, little has been explored regarding the use of worked examples. Silva et al. (2019) proposed a model to guide Software Engineering instructors to create worked examples for teaching UML class diagrams. The objective of the work was to unite the worked examples with open source projects. The authors created guidelines to help faculty select projects and create worked examples drawn from those projects. However, the main focus of the present work is on the selection of OSS projects, and the use of worked examples was little explored.

We can note that the worked examples can be used in different areas, including Computer Science, and can present several benefits for teaching, as reported in the works discussed in this section. However, the use of worked examples in Software Engineering was little explored, making an investigation into the use of this material in this area important.

### 3 Related Work

The use of OSS projects shows benefits for teaching SE, enabling students to develop important skills and experiences necessary for the industry, in addition to providing contact with realistic development environments (Pereira, 2021; Montagner and Kurauchi, 2022; Pinto et al., 2017). There are several studies that seek to improve the teaching of Software Engineering through the use of OSS projects. These studies present different ways to use OSS projects in teaching.

Müller et al. (2019) argue that the use of OSS projects in teaching can be positive and beneficial for students, having a positive impact on students' future careers, as it puts them face to face with real environment problems. This approach can also bring benefits for instructors, as it provides direct access to real problems for possible research projects. However, they can present some challenges, mainly related to the organizational point of view (providing structure, guidance, and control of student involvement), given that this approach requires additional resources in relation to the traditional configurations of the courses.

Some studies (Morgan and Jensen, 2014; Papadopoulos et al., 2013; Nascimento et al., 2018) discuss students' experiences when they have direct contact with OSS projects. The usage of OSS projects proved to be an interesting approach. However, some challenges were encountered, such as the complexity of the projects and the difficulty of the instructor in following the students involved in OSS projects.

Dorodchi and Dehbozorgi (2016) focus on the collaborative skills gained by students when they are exposed to humanitarian OSS projects. Ellis et al. (2008) and Ellis et al. (2015) also addressed humanitarian OSS projects in software engineering education. The use of these projects brought benefits in teaching, but also some challenges, such as the inexperience of students, limited duration of the course, and complexity of the projects (Ellis et al., 2008).

Pereira (2021) work described a practical experience in which students made contributions to OSS projects as part of their final project. The students were assisted by instructors in selecting the projects and had to complete the development or correction of three issues. Among the benefits highlighted by the author of the work are learning version control systems, exposure to quality control, and the experience of dealing with large projects with complex code bases. Additionally, Pereira (2021) emphasizes that contribution tasks in OSS projects can also help instructors, freeing them from code reviews and student monitoring tasks. Among the challenges faced by students were cited understanding the code structures of open-source projects and locating specific parts of the code related to errors and functionalities.

In Montagner and Kurauchi (2022), students were also able to contribute to OSS projects, focusing on building practical development skills within an advanced SE course. The study analyzed students' contributions in terms of type of contribution, code complexity, approval rate, and project size. The authors concluded that the course achieved its goal of providing practical software development skills, and students agreed on the usefulness of the content for future industry work. Furthermore, accepted contributions to well-known projects reinforced the effectiveness of the course.

Just like in the works of Pereira (2021) and Montagner and Kurauchi (2022), Tan et al. (2021) work also provided students with the opportunity to engage with and contribute to OSS projects. In Tan et al. (2021) experiment, students fixed bugs in Java projects available on GitHub and utilized automated program analysis tools. Similarly to the other works, the results show that students perceive improvements in their skills and application of knowledge.

Silva et al. (2020b) work had a different focus from the others, being aimed at teaching UML through OSS projects. The work presents an experience report on the pedagogical use of OSS projects in a Software Engineering course. Such experience took place in the context of isolation imposed by the pandemic of COVID-19, with the course being conducted entirely virtual, with the support of some platforms such as Google Meet, Classroom, GitHub, Padlet and Trello. The projects were used in software modeling activities, which involved UML diagrams.

In the study of Silva et al. (2020b), the students chose the projects they wanted to work on, always taking into account the prerequisites established by the instructor, such as programming language, software size, and number of releases. With the selected projects, the students did the work management, characterized the selected project and created the UML models. The results of Silva et al. (2020b)'s work showed that students agreed that the teaching method used addressed the levels of knowledge, understanding, and application in a satisfactory manner. In addition, they were satisfied with working with real projects, and enjoyed the activities and tools used in class.

The works of Silva et al. (2020a) and Pereira and Díaz (2022) focused on a specific aspect of OSS project usage, which is the selection of suitable projects. Silva et al. (2020a) presented FlossSearch.Edu, a tool that supports students searching for appropriate projects to work on Lessa and von Flach G. Chavez (2020), based on the combination of technical and social criteria. Lessa and von Flach G. Chavez (2020) evaluated this tool from the students' perspective. They found that the tool played an important role in the project selection process, and most students stated that the tool was useful, easy to use and intended to use it in the future, and can be a strong ally in the selection of projects, both for students and instructors of Software Engineering. A tool similar to FlossSearch.Edu was developed in the work of Pereira and Díaz (2022), also focusing on assisting in the selection of OSS projects for students to work on. The developed tool was GitMate, a recommendation system based on GitHub, where students can search for three projects according to the characteristics they desire. According to Pereira and Díaz (2022), the results of using GitMate showed that the tool can help students make comparisons to choose OSS projects.

The aforementioned studies show that the use of OSS projects brings benefits to Software Engineering teaching, mainly related to real learning experiences. However, despite the increased use of OSS projects in education, the choice of appropriate projects, limited course duration, the time and effort invested in planning, and the way to conduct classes, are challenges that can hinder adoption of such an approach by instructors (Pereira and Díaz, 2022; Pereira, 2021; Silva

et al., 2020a, 2019; Raj and Kazemian, 2006; Jaccheri and Osterlie, 2007; Ellis et al., 2008; Nandigam et al., 2008; Morgan and Jensen, 2014). In addition, most of the works that explore the use of OSS projects in Software Engineering education are experience reports.

OSS projects contain several artifacts that can serve as a basis for creating worked examples, to explain certain concepts or techniques of Software Engineering. In the work of Silva et al. (2019) the use of worked examples in conjunction with OSS projects allowed alignment of theory and practice, taking real examples to students. However, the work of Silva et al. (2019) addressed a very specific topic in Software Engineering, the teaching of UML diagrams, in addition to focusing only on the initial recognition of the project.

Taking into account that most instructors are not experts in OSS projects, they may face difficulties in finding worked examples that can be useful in the classroom. Thus, promoting the development of a portal of worked examples extracted from OSS projects, appropriate for the teaching of Software Engineering, has the potential to encourage the adoption of the use of OSS projects in courses, assisting instructors in the search for material and in preparation of classes. Furthermore, the portal can enable teachers to bring real-life examples based on OSS projects into the classroom, providing students with exposure to real-world projects and issues, eliminating the difficulty of choosing a suitable project, and facilitating the understanding of the context in which the example occurs.

In a previous work (Tonhão et al., 2020), the creation and evaluation of a template for creating and structuring worked examples extracted from OSS projects were explored. It is worth noting that the activities reported in the article in question refer to the first Design Cycle presented in this work, and were divided into three stages related to the elaboration and evaluation of the template for cataloging the worked examples.

## 4 Method

This work aimed to reduce the obstacles faced by instructors in the use of OSS projects in the educational environment and in the use of worked examples in the teaching of Software Engineering. For that, we developed an open portal for cataloging worked examples extracted from OSS projects. We adopted the Design Science Research (DSR) research approach, which focuses on the connection between knowledge and practice, emphasizing that it is possible to produce scientific knowledge by designing useful artifacts (Wieringa, 2009).

The DSR paradigm seeks to create new and innovative artifacts that solve real problems in specific domains (Hevner and Chatterjee, 2010; Hevner et al., 2004). Such an approach comprises three research cycles: Relevance Cycle, Design Cycle, and Rigor Cycle. We use other research methods to conduct the steps performed during the execution of the DSR cycles, namely: study of the literature, questionnaire survey, focus group, lab studies with think-aloud protocol, and interviews with instructors.

Each research method serves different purposes and pro-

vided unique insights. The literature study was useful for understanding the existing knowledge and theories related to the concepts addressed in this work. Focus groups and interviews allowed for a detailed exploration of participants' perspectives and experiences, while studies utilizing the think-aloud protocol offered insights into participants' mental processes during specific tasks.

We provide more details for each cycle in each specific section. Figure 1 shows a summary of the DSR applied in this study, laying out its cycles.

In the **Relevance Cycle**, we define the problem to be addressed, the form of investigation, and the requirements for the research (Hevner, 2007). In this research, the problem addressed is related to the difficulties faced by instructors in using OSS projects as material to support the teaching of Software Engineering.

For a better understanding of the problem addressed, we reviewed the literature looking at studies related to the use of OSS projects for teaching Software Engineering. By analyzing the literature, we raised the main difficulties in adopting this type of project in the classroom. We also conducted a survey with 20 instructors (Subsection 5.2) to investigate how they use examples in Software Engineering courses and the difficulties they face when looking for examples.

Based on the organization of information from the literature and the survey analysis, we defined the requirements to design the portal to catalog worked examples for Software Engineering courses. Therefore, we started the first Design Cycle of our proposed artifact.

In the **Design Cycles**, we conducted activities related to the construction of the artifact, its evaluation, and the feedback collection to refine the artifact. This cycle is central to any research project in DSR (Hevner and Chatterjee, 2010). In this work, the proposed artifact was a portal for cataloging worked examples extracted from OSS projects. The goal is to assist instructors who want to adopt OSS projects as teaching support material, offering worked examples that address different topics of Software Engineering.

In this work, we carried out six design cycles, with the participation of 19 instructors throughout the cycles. We invited instructors who worked with the topic to be addressed in this study. In each cycle, a specific group was selected. The choice of participants for each cycle was made taking into account their experience and areas of study, in order to obtain feedback from different profiles. Table 1 shows the information about the instructors participating in each cycle (except cycle 2, which did not involve external members). The identifiers assigned to each instructor (PX) will be used in the rest of the paper to facilitate the identification throughout the text.

The design cycles were divided into two stages. The first stage focused on elaborating and evaluating the model proposed to catalog the worked examples. Before defining the organization of the portal, it was necessary to draw the representation of the worked examples and structure the information needed for the catalog. We split the process of creating and evaluating the model into three cycles, as shown in Figure 2, and composed the first stage of design cycles.

In the second stage, we conducted design cycles with Software Engineering instructors who interacted with the portal,

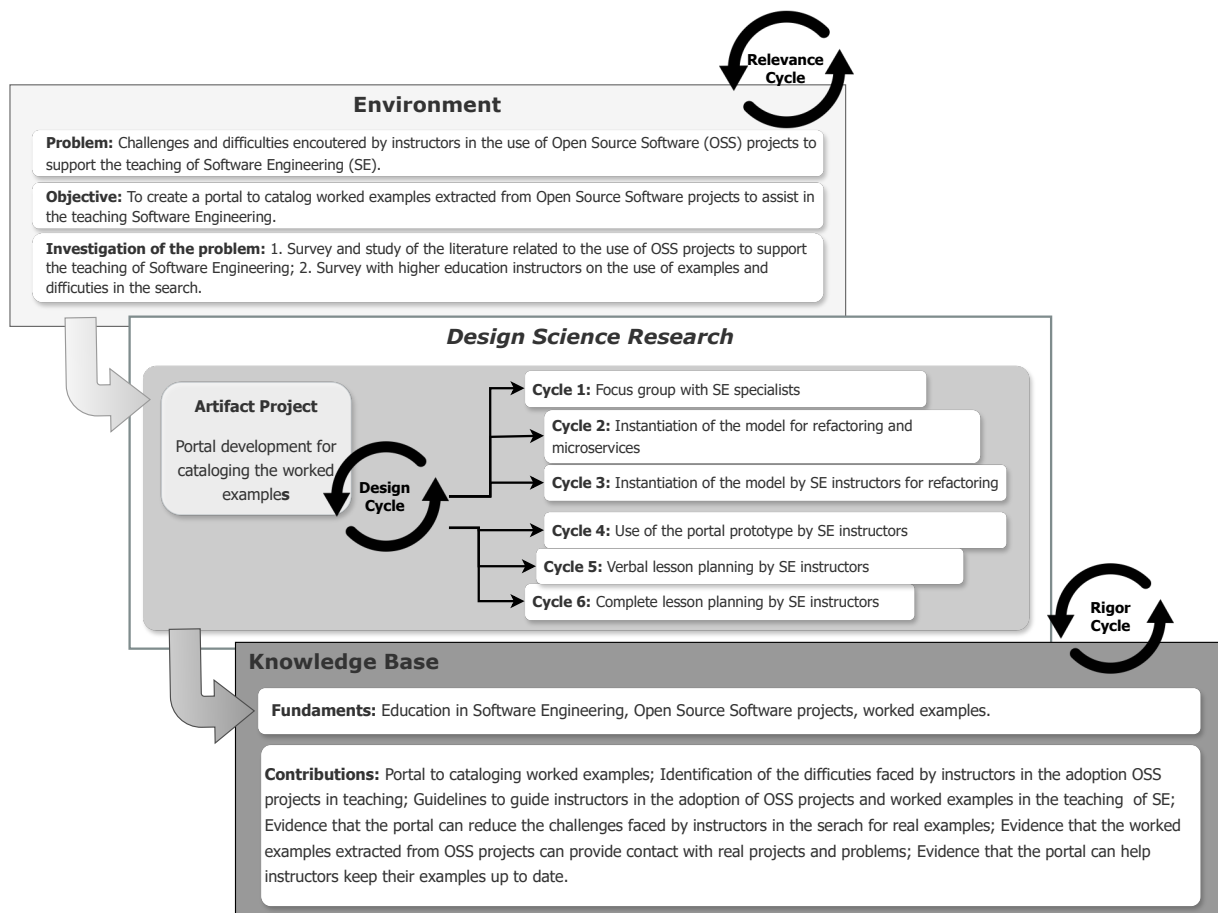


Figure 1. Overview of Design Cycles Applied to Research.

planning the classes in which they would use the examples. The goal of the second stage was to evaluate the usefulness of the portal as a support material in higher education. This stage involved three cycles, which are described in Figure 3.

The details about the methods followed in each design cycle are presented along with the results of the respective cycle.

## 5 Relevance Cycle

We split the Relevance Cycle into two steps. In the first step, we investigated the literature to identify evidence of problems that instructors face when adopting OSS projects in the educational environment. In the second step, we administered a survey questionnaire to investigate the use of examples in Software Engineering courses and the difficulties faced by instructors in using such examples. We detail both steps below.

### 5.1 Study of Literature

To explore how OSS projects are used in the context of Software Engineering teaching, we conducted an ad-hoc exploratory review of literature. This study aimed to detect the

benefits and disadvantages of using OSS projects as an alternative in the teaching of Software Engineering.

Initially, we searched for publications that dealt with the subject of interest. The string used in the search was as follows: “open source projects” OR “open-source software projects” AND “teaching software engineering” OR “software engineering education”. The search was conducted on Google Scholar, which indexes a wide variety of databases, and publications were searched up to the year 2023. After selecting the studies, we read the titles and abstracts, in order to discard articles that did not address the use of OSS projects in the context of teaching Software Engineering. Then, we read the other papers in full aiming to identify the strengths and weaknesses of the adoption of OSS projects in Software Engineering courses, as well as the difficulties faced by the instructors when preparing their classes based on these projects.

We found twelve studies (Pereira and D az, 2022; Pereira, 2021; Silva et al., 2020b; Pereira and Pitxitxi, 2020; Silva et al., 2019; Pinto et al., 2017; Smith et al., 2014; Morgan and Jensen, 2014; Gehringer, 2011; Ellis et al., 2008; Jacheri and Osterlie, 2007; Buchta et al., 2006) that fit the scope established for the search. One author performed the data extraction process and organized it in a spreadsheet with the following information: title, authors, year of publication,

Instructor	University	Cycles					SE Teaching	Use of OSS
		1	3	4	5	6		
P0	PUC-Rio	x		x	x		15 years	Research
P1	UFPA	x	x				-	Research
P2	UTFPR	x	x	x	x	x	14 years	Research/ contribution
P3	UNESPAR	x	x	x		x	5 years	Research/ contribution
P4	UTFPR		x		x		20 years	Research
P5	UEMS		x				5 years	Use only
P6	UTFPR		x				13 years	Research/ contribution
P7	NAU			x			18 years	Research
P8	UFBA			x			16 years	Research/ contribution
P9	UFMS				x	x	2 years	Research/ contribution
P10	UNESPAR				x		17 years	Research
P11	UFU				x		20 years	Research
P12	USP				x		22 years	Research/ contribution
P13	UNIT				x		22 years	Research/ contribution
P14	UENP				x		15 years	Research/ contribution
P15	UTFPR				x		2 years	Research/ contribution
P16	UFABC					x	3 years	Use only
P17	USP					x	15 years	Research/ contribution
P18	UFBA					x	7 years	Research/ contribution

Table 1. Instructors participating in the studies of Design Cycles.

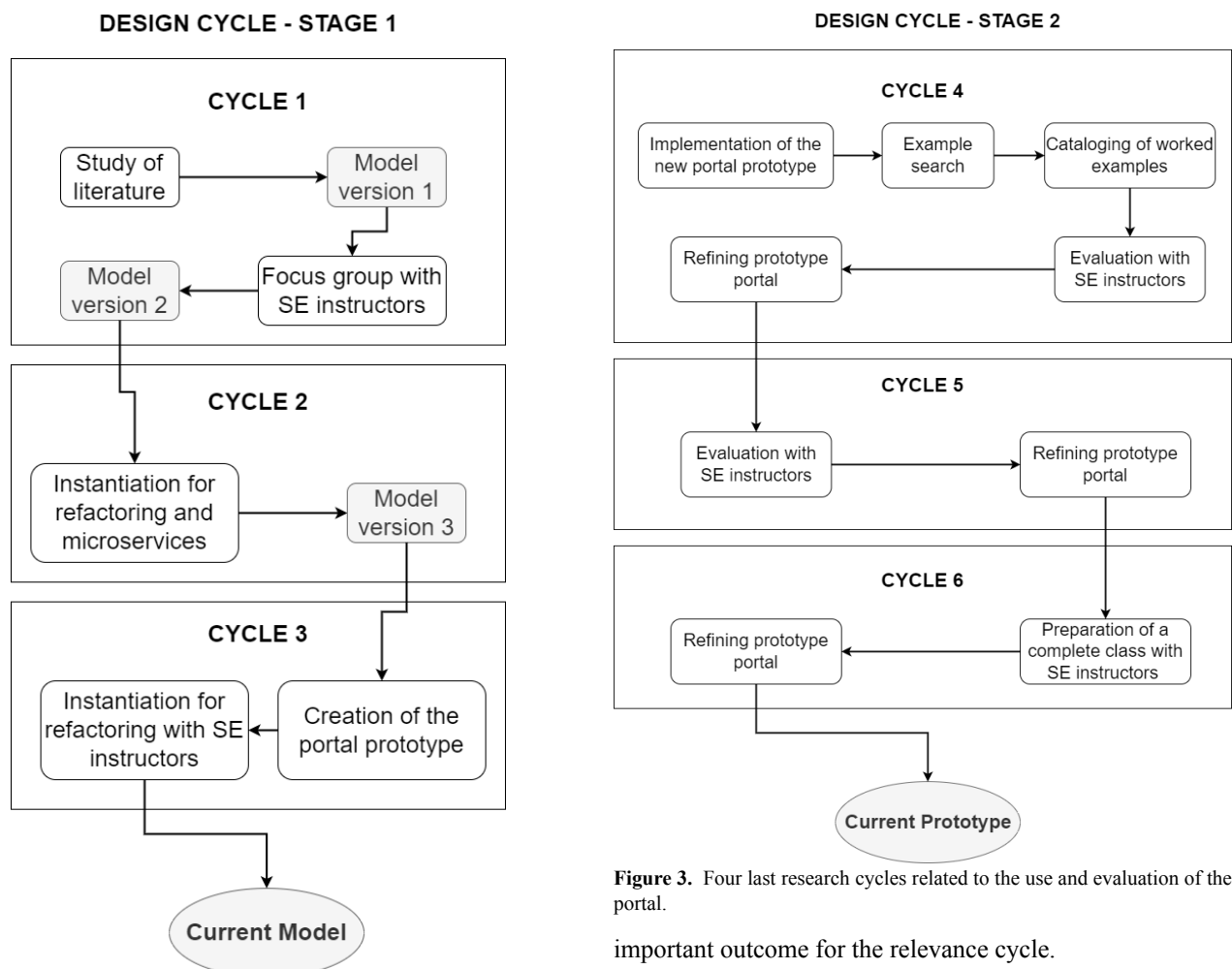


Figure 2. First three design cycles related to creating and evaluating the worked examples cataloging template.

and publication venue. In addition, we used two columns to collect the strengths and weaknesses of using OSS projects in Software Engineering education. The data obtained in this extraction were validated with the other authors during weekly meetings devoted to this. In this paper, we keep the analysis restricted to the weaknesses, given that this is the

Figure 3. Four last research cycles related to the use and evaluation of the portal.

important outcome for the relevance cycle.

The main difficulties identified in the publications were: **choosing appropriate projects** (Pereira and Díaz, 2022; Pereira and Pitxitxi, 2020; Silva et al., 2020b, 2019; Pinto et al., 2017; Smith et al., 2014; Gehringer, 2011; Ellis et al., 2008; Jaccheri and Osterlie, 2007); **limited course duration** (Pereira and Pitxitxi, 2020; Pinto et al., 2017; Ellis et al., 2008); **and the time and effort invested in planning and following the lessons** (Pinto et al., 2017; Morgan and Jensen, 2014; Buchta et al., 2006). These are challenges that could

prevent instructors from taking OSS projects to the educational environment.

Pinto et al. (2017) and Pereira and Pitxitxi (2020) evidenced that the *time limitations* of the course and the *difficulties of choosing the project* are disadvantages associated with the use of OSS projects. Ellis et al. (2008) also pointed out that the *time limitation* can be a challenge, especially when the goal is to try to get students engaged.

Another point mentioned by Pereira (2021), Jaccheri and Osterlie (2007), Ellis et al. (2008), Gehringer (2011), and Smith et al. (2014) was the *difficulty to find projects with appropriate complexity*, which can make it difficult for the instructors and the students to understand. For Smith et al. (2014) the burden of selecting suitable projects can be an impediment to integrating OSS projects and teaching in Software Engineering.

In addition, *the effort to plan and monitor the classes* was mentioned as another challenge (Buchta et al., 2006; Morgan and Jensen, 2014; Pinto et al., 2017), as it may require several hours of the instructor during the week. The need to become familiar with the OSS project can be part of this difficulty since the information in some communities is not organized in a way that facilitates the understanding (Pinto et al., 2017).

Therefore, it is noticeable that the adoption of OSS projects in the teaching of Software Engineering requires some attention, given the challenges and difficulties that instructors may face. Looking for ways to alleviate these challenges can boost the adoption of these projects in the context of Software Engineering courses, and aid instructors who want to bring real experiences and problems to the classroom.

## 5.2 Survey with instructors of Software Engineering

We administered a survey aiming to explore the use of worked examples in Software Engineering courses. We also investigated the possible difficulties that instructors may have in the search for this type of material. The planning was based on studies from the book "Survey Methodology" by Yaacoub et al. (2004), and a pilot study was conducted with two professors to analyze points such as: Clarity and precision of terms, number of questions, format, and order of questions.

The population defined for the study was instructors who teach Software Engineering courses in higher education. The initial sample was defined based on our contacts with individuals belonging to that population. With the initial set defined, the sampling process followed a snowballing process, that is, the individuals initially selected indicated other instructors from their contact networks. We kept the questionnaire open for two months.

We designed the questionnaire to understand if the instructors use examples in their courses and the difficulties in finding them. The questionnaire was composed of twelve questions, related to the respondent's profile and the use of examples, including multiple-choice and essay questions. The questions are presented in Table 2.

### 5.2.1 Survey results

In this section, the discussions of the results obtained in the survey on the use of examples in Software Engineering will be presented. The complete responses are available in a file provided in the repository <sup>2</sup>.

Twenty Software Engineering instructors participated in the research. Six of them hold a Ph.D. degree and 4 have a master degree; fifteen were working in Brazil and five in other countries. Their experience teaching Software Engineering ranged from one year to forty years. Three of the respondents work in the industry in parallel with their teaching assignment, eleven have worked in industry previously, and six have never worked in industry. This information is presented in Table 3.

The answers to the questionnaire showed that all instructors use examples in their courses, 55% of instructors always use them, 40% use them frequently, and only 5% of instructors use them only sometimes. With respect to updating the examples, only 20% of the instructors said they always updated the examples, while the others update them sometimes or never. Interestingly only 10% of the instructors mentioned using **only** real examples; and 35% said that most of their examples are real (see Figure 4).

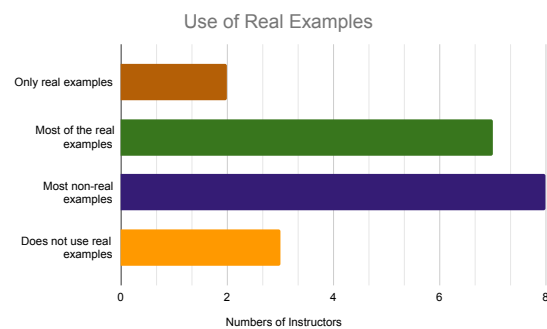


Figure 4. Graph of information about using examples.

When asked where they usually get their examples from (multiple answers allowed), the majority (55%) said that they get the examples from books and the internet, and 35% from real projects available on the internet. The instructors who create their own examples represent 35%, and those who bring experience gained in the industry are 30%. One participant left a comment in the "other" option, in which they mentioned that they usually use open source examples available in scientific papers.

One of the questions in the survey aimed to investigate whether instructor use worked examples in their courses. The term "worked example" was not explicitly mentioned in the question, only the definition was used and the instructors answered whether their examples fit or not. Only two participants stated that their examples never presented a "problem", "steps of the solution", and "result." The others replied that their examples always or frequently contained such elements. Thus, we could notice that most of the instructors in our sample already use, somehow, worked examples.

Another point we investigated was whether the instructors find it difficult to find good examples. Only one participant

<sup>2</sup>[dx.doi.org/10.6084/m9.figshare.25676250](https://doi.org/10.6084/m9.figshare.25676250)

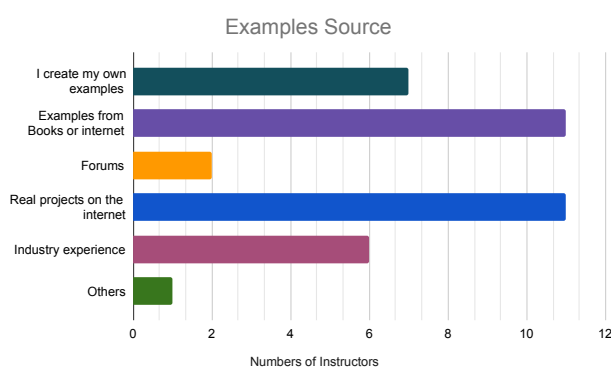


ID	Question	Answer
1	What is your academic background?	Multiple choice [graduate, specialist, master, doctor]
2	How many years have you been teaching Software Engineering?	Numeric response
3	Which courses do you usually teach?	Essay
4	In which country are you currently working?	Essay
5	In addition to teaching, do you work in the software industry?	Multiple choice [I currently work, I have worked, I have never worked]
6	Do you use examples in your lectures?	Multiple choice [Scale: Always - Never (5 levels)]
7	Do you use real examples (from software projects) in your lectures?	Multiple choice [only real examples, most real, most non-real, only non-real]
8	How often do you update your examples?	Multiple choice [always, sometimes, I don't update, I don't use examples]
9	Where do you usually get your examples from?	Selection box
10	When you use examples, do you state the problem, the solution steps, and the result?	Multiple choice [always contains, often contains, never contains]
11	How often do you face problems to find good examples?	Multiple choice [Scale: Always - Never (5 levels)]
12	When looking for examples, what are the main difficulties encountered?	Essay

**Table 2.** Survey questions applied to Software Engineering instructors.

Academic background	Years of experience teaching SE	Country	In addition to teaching, have you worked in the software industry?
Doctor	40	Brazil	Yes, I work in the industry today
Doctor	5	Brazil	Yes, I have worked in the industry at other times
Doctor	6	Brazil	Yes, I have worked in the industry at other times
Doctor	5	Brazil	Yes, I have worked in the industry at other times
Master	12	Brazil	Yes, I work in the industry today
Master	4	Brazil	No, I never worked in the industry
Doctor	2	Brazil	No, I never worked in the industry
Doctor	20	Brazil	Yes, I have worked in the industry at other times
Doctor	2	Brazil	No, I never worked in the industry
Doctor	18	Brazil	Yes, I have worked in the industry at other times
Doctor	0	India	No, I never worked in the industry
Doctor	10	Brazil	Yes, I have worked in the industry at other times
Master	10	Brazil	Yes, I have worked in the industry at other times
Doctor	12	The Netherlands	No, I never worked in the industry
Doctor	5	New Zealand	Yes, I have worked in the industry at other times
Doctor	1	Germany	Yes, I have worked in the industry at other times
Doctor	16	Brazil	Yes, I have worked in the industry at other times
Master	2	Brazil	Yes, I work in the industry today
Doctor	10	Brazil	Yes, I have worked in the industry at other times
Doctor	9	Canada	No, I never worked in the industry

**Table 3.** Table of information about using examples.



**Figure 5.** Graph of information about location of search for examples.

said that they rarely had trouble finding examples. This instructor works in the industry in parallel. The other participants said they find it difficult to find good examples, either always, frequently, or sometimes.

At the end of the questionnaire, we left an open question in which the participants could describe the difficulties encountered when looking for examples. Among the comments, only one instructor left a positive comment mentioning that they rarely have trouble finding good examples. They use the projects developed by the students as examples for the

next courses, and these projects meet the real demands of the university and the region. The other instructors pointed out some difficulties, as follows: **Complexity of the examples and suitability to the course, difficulty to find examples for different topics and approaches, not knowing where to look for examples, and not having examples from open source.**

Regarding the **complexity of the examples and suitability to the course**, they have mentioned that finding complete examples that suit the topics and content being taught is challenging. Moreover, they said that the examples are often too big to be explained in class, and the small examples are not realistic. Another aspect mentioned by the instructors was **difficulty to find examples for different topics and approaches**. Finding examples for specific topics that are updated according to the new practices, methodologies, and technologies used in the industry can be a big problem, most of the examples found are old and reflect previously used methodologies. Another difficulty was related to **not having examples with source code available/open**. Finally, **knowing where to look for the examples** was mentioned since the instructors are not aware of repositories that provide specific examples for Software Engineering.

### 5.3 Relevance Cycle Considerations

Through the study of the literature, we found evidence of some of the challenges faced by instructors in the adoption of OSS projects in Software Engineering courses. Among them, we highlight **the choice of the appropriate project** and the **time and effort required to plan and monitor the courses**. Such challenges can end up hindering the adoption of OSS projects as an alternative in the teaching of Software Engineering.

Regarding the results obtained in the survey, we noticed that all respondents use examples and that a good part uses worked examples, even if they do not know the term. However, only two instructors answered that they use only real examples, and seven use real examples most of the time. Still, we observed that one of the greatest difficulties in seeking examples is related to aspects of complexity and adaptation to the course. According to the participants, it is **hard to find complete and suitable examples for different topics** and approaches that follow the evolution of Software Engineering. According to Pinto et al. (2017) finding real examples is not an easy task and there is a lack of materials and resources and—as confirmed by our survey—**instructors do not know where to look for examples**.

Therefore, in order to alleviate the challenges in the adoption of OSS projects in the educational environment, and the difficulties encountered by instructors who want to find examples, we propose the creation of a portal for cataloging worked examples drawn from OSS projects. The portal is a repository where the instructors can search for real examples of different OSS projects, making it easier to find examples and choose appropriate OSS projects. Based on the literature and the survey, we have defined the following requirements for this portal:

- **Requirement 1:** The portal should provide guidelines to instructors on how to create and standardize the worked examples extracted from OSS projects.
- **Requirement 2:** The portal should openly provide worked examples to support Software Engineering teaching, aiming to reduce the effort to search the resources to exemplify the contents of the course.
- **Requirement 3:** The portal should guide instructors in applying worked examples when teaching Software Engineering.
- **Requirement 4:** The portal must enable cataloging worked examples that address different topics of Software Engineering.

In Table 4, the relationship between the challenges encountered in the adoption of OSS projects in SE teaching and the requirements defined for the elaboration of the proposal is presented.

## 6 Design Cycle - Stage 1

The first stage of the design involved three cycles, which are related to creating, evaluating, and refining the “cataloging template” worked examples extracted from OSS projects. In the first cycle, we developed the first version of the template

based on the literature. We assessed the feasibility of using the template by means of a focus group with Software Engineering instructors. In the second cycle, we instantiated the template to create worked examples for the topics of refactoring and microservices. The goal was to evaluate the template’s flexibility for different topics. In the third cycle, we conducted a study with Software Engineering instructors, in which they used the template to create worked examples on the theme refactoring.

### 6.1 Cycle 1 - Elaboration and Evaluation of the Cataloging Template

We conducted this cycle following the steps shown in Figure 6. The first step of this design cycle consisted of defining the form of representation of the worked examples. As it is important to make the examples easy to create, understand, use, and adopt, we have developed a template for creating and structuring the worked examples extracted from OSS projects.

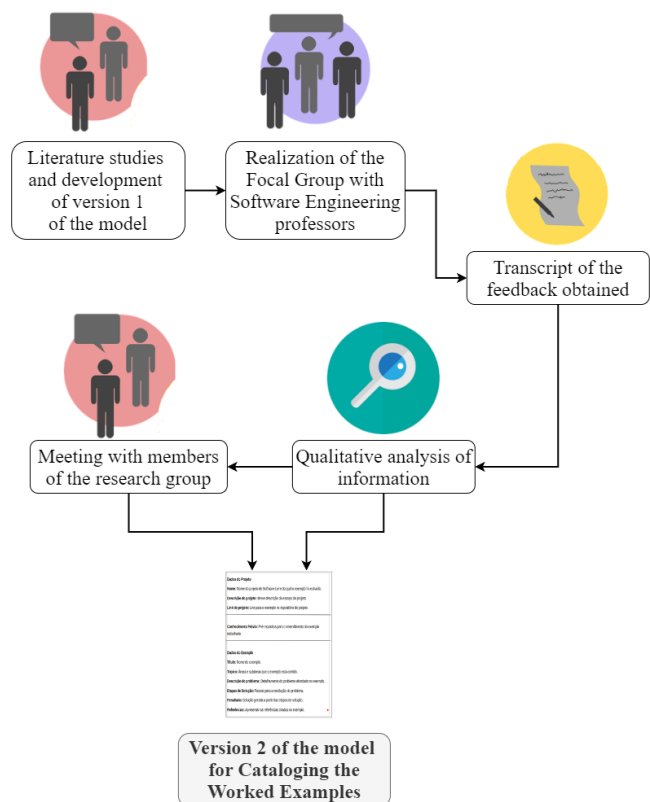


Figure 6. First research design cycle.

We developed the **first version of the template based on the literature** (Silva et al., 2019; McGinn et al., 2015; Booth et al., 2015; Sweller et al., 1998). The focus was to provide guidelines to identify, characterize, and structure the information in the worked examples. The template is presented as a solution of a pattern developed for cataloging the worked examples. The complete pattern can be consulted at <https://portalworkedexamples.herokuapp.com/padrao.php>. This pattern was based on the typical reengineering pattern format described in Demeyer et al. (2008). We present the first version of the template in Figure 7. The template was proposed as the way to catalog the

Challenges	Requirement
Difficulty in finding examples of open-source code.	The portal should provide guidelines to instructors on how to create and standardize the worked examples extracted from OSS projects.
Instructors do not know where to look for examples. The choice of the appropriate project.	The portal should openly provide worked examples to support Software Engineering teaching, aiming to reduce the effort to search the resources to exemplify the contents of the course.
Time and effort invested in planning and following the lessons.	The portal should guide instructors in applying worked examples when teaching Software Engineering.
It is hard to find complete and suitable examples for different topics.	The portal must enable cataloging worked examples that address different topics of Software Engineering.

**Table 4.** Relationship between challenges and defined requirements.

worked examples, divided into three sections: project data, prior knowledge, and example data.

<p><b>Project Data</b></p> <p><b>Name:</b> Name of the open source project from which the example was taken</p> <p><b>Project Description:</b> Brief description of the project scope</p> <p><b>Project Link:</b> Link to the example in the project repository</p>
<p><b>Prior Knowledge:</b> Prerequisites for understanding the example worked.</p>
<p><b>Example Data</b></p> <p><b>Title:</b> Example name</p> <p><b>Topic:</b> Area and subarea that the example is contained in</p> <p><b>Problem Description:</b> Details of the problem addressed in the example</p> <p><b>Solution Steps:</b> Problem solving steps</p> <p><b>Result:</b> Solution generated from the solution steps</p> <p><b>References:</b> Display the references cited in the example</p>

**Figure 7.** First version of the template for cataloging worked examples.

With the first version of the template created, we started the evaluation phase. **The evaluation consisted of a focus group**, based on the five steps presented in Kontio et al. (2004). The focus group was conducted with Software Engineering instructors. The focus group’s objective was to assess the feasibility and get feedback about the proposed template.

In the second stage, we planned the focus group, which would be conducted via videoconference. We also designed the questions used to guide the discussion during the focus group, which are presented in Table 5.

The third step consisted of defining the study participants. It is important to recruit representative, insightful, and motivated participants for the focus group to be successful (Kontio et al., 2004). For that, we selected four Software Engineering instructors, according to the profiles presented in Table 6.

The fourth step was the conduct of the focus group. The focus group session started introducing the project, session rules, and goals. Next, the instructors received the template and a worked example of the refactoring topic, created following the template. To conduct the study, it was necessary to define a topic to be initially worked on, as it would not be feasible to address software engineering as a whole. Therefore, we decided to focus on the topic of refactoring, as there was already a scenario for such a topic extracted from an OSS project.

Subsequently, the instructors received the artifacts and were invited to discuss them. After the instructors discussed the artifacts, questions were asked to continue the discussion and obtain answers about more specific points. The focus group was recorded with the consent of all participants.

After conducting the focus group, the data was **transcribed from the recordings**. Subsequently, **qualitative analysis of the data** was performed, followed by **meetings with the research group members** to organize the information and identify suggested improvements to the template. The results of this cycle are presented in the next section.

## 6.2 Cycle 1 - Results

Through the analysis of the transcripts, we identified candidate improvements in the template. A point made by P0 was the fact that the **structure of the worked example is inflexible**, in their words “*sometimes there are situations when the instructor wants an example just to discuss alternative solutions, [...] and I saw that, when describing the solution, we apparently fix a single solution, and I was wondering if it would not be interesting to leave this open*”.

P2 also mentioned that the structure was a little inflexible, always having to present a single solution. P1 agreed that leaving the structure too strict might make it not fit for examples of some topics. In their words “*examples involving code, [...] examples about tests, maybe it would fit well ..., for a little more general examples, not involving code, it could be a little more difficult*”.

However, P0 came back to argue and realized that the template had some freedom to document more than one solution. For them, “*actually there is some freedom, because there is a sentence in the description saying ‘It is necessary to present all the details of the step by step to reach the final solution’, [...], maybe I can improve the description a little because the step-by-step can be exactly that, for the student to consider different solutions for that problem and at the end, the final solution is provided*”. A possible solution presented by the instructors to solve this problem was to **improve the description of the template**, leaving the possibilities more open, pointing out that nothing prevents the instructor from documenting multiple solutions.

Another problem raised was the **lack of pedagogical elements** in the template, in P2’s words “*I felt lack of pedagogical characteristics, lack of things to link the example with the course or with the specific content of the course [...] Where it could be used and how it could be applied in the classroom*”. P0 completed, saying that the names of the elements were too generic, any Software Engineering model would have these names, but it lacked a more pedagogical language.

P0 pointed out another problem, which was the **lack of more specific elements about open source**, mainly related to the project activity, “*It would be important, as an instructor looking for the examples, to have information if that*

N°	Question
1	Is the idea of creating a template, like the one proposed, to guide the creation and structuring of worked examples interesting? Does it seem feasible?
2	How generic is the template? Are there elements that could be more specific?
3	How clear and consistent is the step by step?
4	Regarding the structure of the example, does the explanation of each defined field present enough information for the example creator to understand the function of each item present in the structure? How could we make it more specific?
5	What would make you use or stop using the proposed template/solution?

Table 5. Questions in the Focus Group.

Instructors	University	SE Teaching	Use of OSS
P0	PUC-Rio	15 years	Research
P1	UFPA	–	Research
P2	UTFPR	14 years	Research/ contribution
P3	UNESPAR	5 years	Research/ contribution

Table 6. Information on participants in cycle 1 studies.

worked example is from an active project, or an active module, that developers are available, maintaining”. The project information is very basic, it would be important to consider other project information that is available at the time of documentation. P1 and P2 also agreed that the **information about the open source project should be more detailed**, indicating exactly where and when the example was extracted.

P1 and P0 raised another problem, the lack of separation between what the problem is and the context, that is, the **lack of an element called context**, before the problem description. P2 also agreed that it would be interesting to have more **information about the context of the example**, but pointed out that it would be important to consider the size of the structure as if it is too long, instructors may not want to catalog the examples, due to the workload required.

Instructors were asked if they would use the template to create examples and if they would use the examples created from the template. For P1 having to create the examples would be a limitation, if they had the examples ready I would be more encouraged to use them, in their words, “*if I had to make the examples it would be a very big limiter, if I eventually had the worked example in the context of my course, already mature enough to use, I would use*”.

P3 would be willing to create the examples and use them because it is a job they already do in their course, so they would feel more encouraged, but they confessed that if there were examples cataloged by other instructors, it would make it much easier. In the words of P3 “*what would most motivate me to use it is that I’ve been working a lot with examples, and what limited me was that the first time I had to build everything [...] so if I already had other examples that other instructors had put in, it would make it much easier*”.

On the other hand, P2, what would attract them would be examples that are simple to understand and with guaranteed pedagogical validity. For P2 “*one of the things I would look to apply would be this question of simplicity and validity, I think these are two points that are important [...] to know which pedagogical applications the example had, and somehow knowing if that had the expected effect, that’s a factor that would motivate me to use*”.

P0 would be afraid to use it if they were not sure where the solution in the example came from, “*I wouldn’t use it if I started noticing that several of the worked examples are artificial solutions created by the instructor, or if I don’t have this information [...] when the solution is not created by someone*

*from the project, starts to lose a bit of meaning, so I would feel discouraged if I notice that in the catalog there are many occurrences of this type*”.

After analyzing all the feedback obtained from the instructors, it was possible to identify points of improvement in the worked examples cataloging template. In this way, the necessary and possible improvements were implemented, and we proceeded to a new design cycle.

### 6.3 Cycle 2 - Creation of worked examples

In Figure 8, the activities developed in the second design cycle are presented. The main goal of this cycle is to verify the flexibility of the template for cataloging the worked examples, for different topics of Software Engineering. The activities of this cycle involved searching and selecting examples related to the topics of refactoring and extraction of microservices and the use of the template for creating examples for these topics. We also analyzed the examples created, to verify the flexibility of the template.

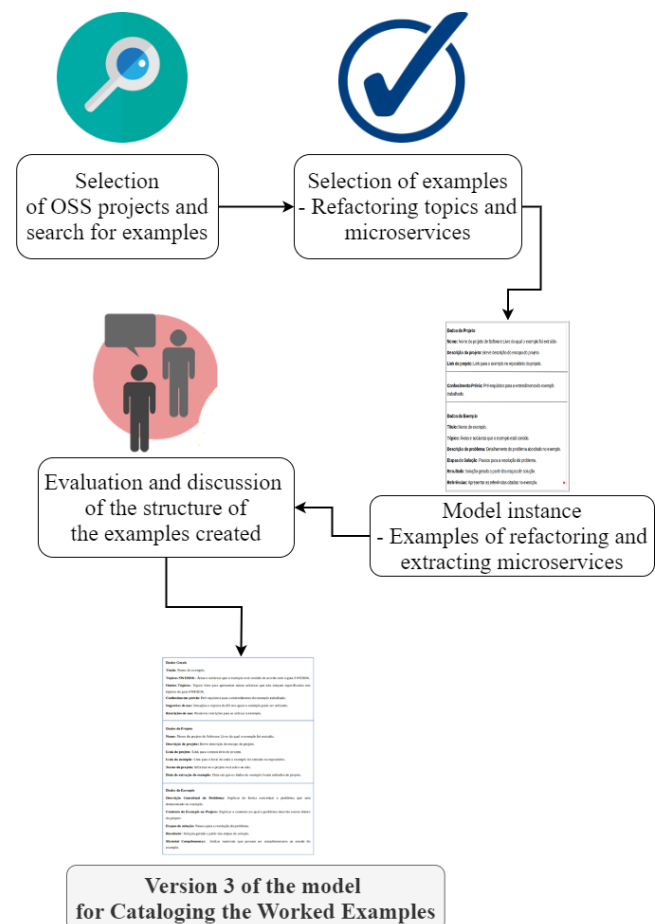


Figure 8. Second research design cycle.

The first step of this cycle was searching and selecting examples. We chose to work with the topics of refactoring and microservices, considering that, one deals with the system at the code level, and the other at the software project level, thus generating examples of different formats. The search was conducted by the members of the research group, looking for pull requests available on GitHub repositories, using the GitHub public API.

The search for the refactoring topic was performed manually. Three OSS projects were selected, namely: Arduino,<sup>3</sup> CodeIgniter 4,<sup>4</sup> and OpenRCT2.<sup>5</sup> We analyzed the pull requests and commits that returned our search using the keywords "refactoring". Three changes related to refactoring had been selected, one from each project. For the microservice extraction topic, we consulted with an expert, who pointed to JPetStore<sup>6</sup>. We selected two repositories from there: one that deals with the system in monolithic form, and the other where the system was organized using microservices.

With the selected examples, the next step was to catalog the worked examples according to the template proposed. For this cycle, the researchers involved in the study created the entries in collaborative sessions, mapping the information available to the template. We created three worked examples for the refactoring topic<sup>7</sup>. Such examples covered the refactoring operations inline method, rename variable and remove redundancy. For the microservice extraction topic we created examples related to the process of extracting microservices from a monolithic architecture, comparing the system in the two selected repositories of the JPetStore.

The worked examples created in this cycle, went through a quality evaluation, to verify the effectiveness of the template in the creation of worked examples, and the adaptation of the template to examples of different topics and formats. The evaluation was conducted by members of the research team, who critically analyzed the resulting examples looking for improvements. The analysis was focused on the fluidity of reading the examples and organizing the information.

In parallel to the creating of the examples, we defined the organization of the cataloging portal preliminary version. For this purpose, we analyzed the Curricular Guidelines for Graduate Programs in Software Engineering, of the ACM (Association for Computing Machinery) (ACM, 2015), and the Knowledge Software Engineering Body, SWEBOK Guide (Bourque and Fairley, 2014). Such documents were analyzed, and topic maps<sup>8</sup> were created in order to discuss the way that the Software Engineering content would be organized. The maps show the topics of the first two levels of each document. By comparing them, we notice that the SWEBOK guide has a greater granularity of topics, presenting a greater number of areas in each level, which would become too hard to navigate. Therefore, we chose to organize the first version of the portal using the SWEBOK guide.

<sup>3</sup><https://github.com/arduino/Arduino>

<sup>4</sup><https://github.com/codeigniter4/CodeIgniter4>

<sup>5</sup><https://github.com/OpenRCT2/OpenRCT2>

<sup>6</sup><https://github.com/mybatis/jpetstore-6>

<sup>7</sup>Available at <https://figshare.com/s/ad87727b39cae679ab7b>

<sup>8</sup><https://figshare.com/s/fe7fba6fc37cc353e966>

With the execution of this cycle, it was possible to identify some points of improvement in the template and define the organization of the portal. In the next subsection, the results obtained in this cycle will be presented.

## 6.4 Cycle 2 - Results

From the execution of the process of creating the worked examples following the proposed template, it was possible to raise some points of improvement. The first problem was related to **fluidity in reading the examples**. There was a context element in the template—in which the context of the example was explained—and a problem description element—which explained the problem addressed in the example. In this way, the information did not seem to complement each other. First, the context of the example in the project was explained, without mentioning the problem, and only later the problem that would be addressed was discussed.

Another problem identified was the **organization of information**. It was identified that improvements could be made in splitting the information in the example and in the order in which this information was presented. A division of sections could help in the organization of information and in the reading flow—with each section presenting a specific focus, such as project data, and example data.

Regarding the definition of the portal organization for cataloging the worked examples, after analyzing the maps generated from the selected documents, it was concluded that the SWEBOK Guide would be more suitable as a basis for cataloging the worked examples. This guide was chosen because it is more granular and the topics are better defined, making it easier to guide users of the portal in the search for worked examples for more specific contexts.

Furthermore, the examples from this study were cataloged using *tags*, with the aim of making the topics related to the example more specific. The initial *tags* database was extracted from the Stack Overflow data *dump*, made available through the SOTorrent dataset (Baltes et al., 2018). We used these tags as a seed. The idea is that instructors add new *tags* to the database when needed—whether related to technical or pedagogical attributes.

## 6.5 Cycle 3 - Use of the template by Software Engineering instructors

In Cycle 3, the objective was to evaluate the instantiation of worked examples using the template. For that, we created a prototype of a portal based on the cataloging template. The instantiation was performed by a group of Software Engineering instructors. In Figure 9, we outline the activities conducted in Cycle 3.

For Cycle 3, we selected another set of examples from OSS projects, to complement the search carried out in the previous cycle. In this cycle, we focused only on refactoring to keep the environment more controlled. The research group discussed a set of pull requests obtained via GitHub API, exactly as we conducted in Cycle 2. The six scenarios selected in this phase were taken from the following projects:

Arduino, OpenMRS<sup>9</sup> and Apache Dubbo<sup>10</sup>.

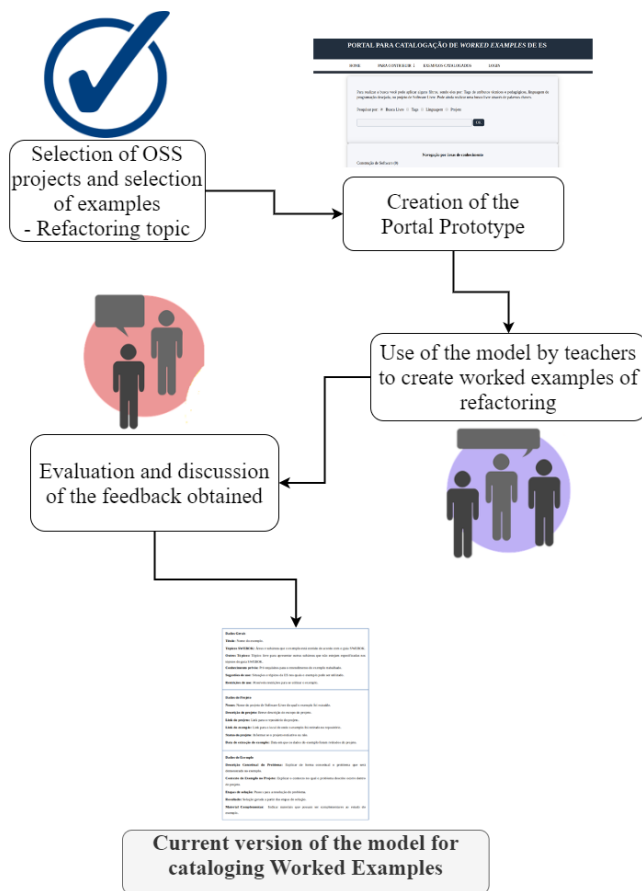


Figure 9. Third research design cycle.

Then, we invited six Software Engineering instructors—P1–P6 (Table 7)—to use the template and create the worked examples for the selected scenarios. Each instructor received one refactoring example, the template for cataloging, and a link to the portal prototype.

Instructor	University	Teaching SE	Use of OSS
P1	UFPA	—	Research
P2	UTFPR	14 years	Research/ contribution
P3	UNESPAR	5 years	Research/ contribution
P4	UTFPR	20 years	Research
P5	UEMS	5 years	Use only
P6	UTFPR	13 years	Research/ contribution

Table 7. Information on participants in cycle 3 studies.

We collected data by following a think-aloud protocol, via videoconference. In this method, the participant is asked to speak aloud during the execution of a task or to solve a problem (Jaspers et al., 2004). During the process of creating the worked example, the instructor verbalized their thoughts and raised possible problems and improvements in the template. We also took notes of the participants interactions for further analysis.

We conducted each session individually. Each participant received a link to a page with instructions, the template and a form for creating the example. They also received a link to the pull request in which the refactoring was performed.

The instructors were invited to analyze the pull requests, and based on the template, create and catalog the worked example from the form.

After each session, we analyzed the actions, notes and verbalizations. The members of the research group discussed them in order to understand the drawbacks and the necessary improvements.

## 6.6 Cycle 3 - Results

Based on the sessions, we identified some points of improvement and adjustments in the template for creating the worked examples were raised. First, some instructors **felt confused about the “usage restrictions” element**, P2 found the element unnecessary as they could not find any restrictions for using the example they were creating. P5 found it too similar to the “prior knowledge” element, since the prior knowledge is already a restriction of use.

Another point raised by P2 was the **lack of awareness for the mandatory elements** in the structure of the example, besides suggesting that some elements could be mandatory, such as “Previous knowledge”, “Suggestions for use”, and “Result”. P4 also pointed out that for them “Prior knowledge” should be a mandatory element.

Thinking about focusing only on essential elements that make a difference in the structure, P1 argued that instructors may **not understand the “project status” element**. Given that this is a non-core element, perhaps it could be removed. P2 suggested **clarifying what the “References” element is**, or naming it with a more explanatory name, as the way it was described is not explicit which types of references should be placed in the element.

P2 also found it **confusing to have fields to place images in all elements** of the data section, so they could only have the problem and the result. P1 and P4 also preferred not to add images, as the ideal would be for students to look at the example link, and see the problem actually in the project and not in an image extracted from it. P6, on the other hand, thought the idea of images was good because if the example link eventually disappears, the images can be a way for the examples to remain usable.

With Cycle 3, the stage 1 of the *design* cycles was concluded, which addressed the elaboration and evaluation of the standard for cataloging the worked examples. In the next section, a discussion of the results of this stage will be presented.

## 6.7 Results and Considerations - Stage 1

We implemented improvements in the template at the end of each cycle, always taking into account the feedback received. In Cycle 1, one of the main problems identified was the lack of pedagogical elements in the template. To solve this problem, we created two pedagogical elements in the template: “Usage Suggestions” and “Usage Restrictions.” This would assist instructors in the selection of the application of the examples.

P1 and P2 suggested the creation of an element to separate the context and the problem since they missed details of the

<sup>9</sup><https://github.com/openmrs/openmrs-core>

<sup>10</sup><https://github.com/apache/dubbo>

context of the project in which the example occurs. We created a new element called “context” to enable instructors to describe this context.

Another problem pointed out was the lack of more elements related to OSS projects: link to the project repository (P2) link to the example (P2), project status/activity (P0, P1)—active or inactive project—and the extraction date example (P1). Therefore, we created the elements “example link”, “project status” and “extraction date”. In addition, we define and marked the mandatory elements in the structure of the worked examples.

Such improvements directly impacted the **Requirement 1**—guidelines to guide instructors in the creation and cataloging of examples. The clearer and more cohesive the information is presented, the easier the process of creating the example is. The application of improvements in this cycle generated the second version of the template for cataloging the worked examples on the portal, which served as an entry for Cycle 2 of evaluation.

In Cycle 2, the changes were made mainly to improve the fluidity in reading the examples. The order of the “context” and “problem description” fields has been changed, and the elements have been renamed to: “conceptual description of the problem” and “context of the problem in the project”. In addition, we explicitly split the template into three sections: general data, project data, and worked example data, so that the information was better presented and divided.

The *general data* section presents information related to the worked example itself (e.g., title), and pedagogical elements (e.g., prior knowledge and suggestions of use). In the *project data* section we present the project name, links to the repository, example covered, and some additional information such as description, project status, and date of the sample extraction. The *worked example data* is based on the definition of the worked example, with elements such as problem description, the context of the problem in the OSS project, steps of the solution, and result.

After Cycle 2, we have also included two new elements in the template, in order to guide instructors in the search for examples. The first was the topics of SWEBOK to which the example is related to—these topics are already registered and the instructor only needs to select them. The second element inserted was a set of tags, in which the instructor is allowed to label the example with technical and pedagogical attributes (choosing a previously recorded tag, or creating new ones).

The improvements made in Cycle 2 are also related to **Requirement 1**. In addition, they are related to **Requirement 4**—the portal should allow the cataloging of examples for different topics. Cataloging according to SWEBOK topics and tags will help the instructor guide their research according to the desired topics and will allow the portal to be organized according to a topic outline.

In Cycle 3, we added a new open element called “other topics”, to complement the topics of the SWEBOK guide. In addition, the “references” field of the worked example has been renamed “complementary material”, and the description of the element “tags” has been improved. This improvement in the description allows instructors to understand that they can insert up to three tags with technical and pedagogical attributes related to the worked example and that new

tags can be created.

We changed the fields “prior knowledge” and “restrictions on use” (created in cycle 1), into a single element, for presenting similar information. In addition, the fields “prior knowledge and usage restrictions”, “suggestions for use” and “result” of the worked examples were made mandatory, as they were considered essential elements to catalog the worked examples. The element “project status” (created in Cycle 1) was removed since it was not considered important given the existence of the other elements from where it is possible to infer the status.

We consider that the portal met **requirement 1** at this point: provide guidelines to guide instructors in the creation and standardization of the worked examples. The updated template is presented in Figure 10, and served as the basis for creating a new prototype of the portal<sup>11</sup> used in from this cycle on.

<p><b>General Data</b></p> <p><b>Title:</b> Name of the example.</p> <p><b>SWEBOK Topics:</b> Areas and subareas that the example is contained in according to the SWEBOK guide.</p> <p><b>Other Topics:</b> Free topic to present other subareas that are not specified in the SWEBOK guide topics.</p> <p><b>Tags:</b> Technical or pedagogical tags related to the worked example.</p> <p><b>Prior Knowledge and Usage Restrictions:</b> Prerequisites for understanding the worked example, and possible restrictions for its use.</p> <p><b>Usage Suggestions:</b> Present information on how to use, when to use, and with whom to use the worked example.</p>
<p><b>Project Data</b></p> <p><b>Name:</b> Name of the Free Software project from which the example was extracted.</p> <p><b>Project Description:</b> Brief description of the project scope.</p> <p><b>Project Link:</b> Link to the project repository.</p> <p><b>Example Link:</b> Link to the location where the example was taken from the repository.</p> <p><b>Extraction Date:</b> Date when the sample data was taken from the project.</p>
<p><b>Worked Example Data</b></p> <p><b>Conceptual Description of the Problem:</b> Explain in a conceptual way the problem that will be demonstrated in the example.</p> <p><b>Context of the Project Problem:</b> Explain the context in which the problem described occurs within the project.</p> <p><b>Solution Steps:</b> Steps to resolve the problem.</p> <p><b>Result:</b> Solution generated from the solution steps.</p> <p><b>Complementary Material:</b> Indicate materials that can be complementary to the study of the example.</p>

Figure 10. Current Worked Examples Cataloging Template.

## 7 Design Cycles – Stage 2

The objective of Stage 2 was to evaluate the usefulness of the portal as a support material for Software Engineering courses. To do so, we cataloged refactoring examples and carried out evaluations with Software Engineering instructors to get feedback about the portal and the examples to prepare the lectures. This phase had 3 cycles (Cycles 4, 5, and

<sup>11</sup><https://portalworkedexamples.herokuapp.com/>

6), all of them conducted with Software Engineering instructors.

## 7.1 Cycle 4 - Use of the portal prototype by SE instructors

In this cycle, we developed and made available the first version of the portal. An impression of the search page for this first version is shown in Figure 11.



Figure 11. Search page for the first version of the portal.

For the first round of evaluation collected more examples and instantiated them in the portal using the template. We chose to select examples for the topic refactoring because it has already been worked on in other phases of the research. We surveyed examples using the GitHub API. The search performed returned pull requests related to the searched keywords<sup>12</sup>. In our case, the keywords used were “refactor” and the names of refactoring types (e.g, extract method, extract variable). As a selection criterion, we considered the ease of understanding of the problem addressed, the number of changes made, and the classes involved in refactoring. We chose to select simpler examples, which did not involve very complex refactorings. We ended up selecting 11 examples.

We also posted a question on social media (Twitter and Facebook) in order to collect examples from the context of OSS projects. Some users commented on the publication, one of them suggested using the Why Refactoring<sup>13</sup> website, which contains a refactoring data set carried out on projects on GitHub (Silva et al., 2016). We have selected nine examples available on the website. Finally, another set of ten examples was made available by master’s and doctoral students at PUC-Rio, who research the topic in a renowned laboratory.

We cataloged the thirty selected examples on the portal. The next phase consisted of inviting instructors to participate in the portal’s first assessment. The invitation was made to five instructors who agreed to participate in the study. Three of them had already participated in previous stages, and two had no knowledge about the project. In all cycles, we invited some instructors who took part in this research in a previous cycle and others who were not aware of the project, to receive feedback from different points of view. The profile of the participants in this cycle is shown in Table 8.

Once again, we applied a think-aloud protocol, now followed by a debrief focused on understanding how they would use the portal and how they would apply the worked examples in class. The questions asked in the debrief are presented

Instructor	University	SE Teaching	Use of OSS
P0	PUC-Rio	15 years	Research
P2	UTFPR	14 years	Research/ contribution
P3	UNESPAR	5 years	Research/ contribution
P7	NAU	18 years	Research
P8	UFBA	16 years	Research/ contribution

Table 8. Information on participants in cycle 4 studies.

in Table 9. In a synchronous videoconference session, each instructor was invited to use the portal and look for examples to use in a refactoring lecture, select some of the examples, and evaluate them. Each instructor narrated the search on the portal and made suggestions for improvements to the cataloged examples and the navigability of the portal. At the end of the process, we proceeded with the debriefing session.

After the evaluation with all the instructors, we analyzed the data collected and categorized the feedback obtained. Subsequently, we discussed the feedback as a team and implemented the necessary changes on the portal.

## 7.2 Cycle 4 - Results

In general, the improvements pointed out in this cycle were related to **structure of cataloged examples**, **portal usability**, and **search in the portal**. Regarding the **structure of the examples**, P2, P3, and P8 suggested the insertion of a “programming language” as a mandatory element to the example. For P3, “*the instructor could choose the examples taking into account the project and the programming language as well*”.

Regarding **portal usability**, P7 and P8 pointed out that the project and example links were not clickable links (the instructor had to copy and paste the link to navigate). Another suggestion made by P7 was for the *tags* and *topics* to be clickable too, bringing other examples cataloged with those items, facilitating the search for related examples.

In addition, some instructors suggested to **expand the example search** and make it more flexible, P7 suggested searching the example description, and P0 suggested “*more flexibility in the search, other fields than the predefined ones, search through from other aspects, free form, maybe a free search box*”. P2 also pointed out that the search should be more flexible, P3 and P0 suggested searches by project and by programming language.

During the debrief, we asked some pedagogical questions related to the use and usefulness of the examples used in teaching (9). The results of such questions are presented below.

All instructors participating in the study found ways to use the examples in class, some using more traditional methodologies, and others focusing on active methodologies. P0 would approach the example more actively, first showing the classes involved in the example, and asking the students to try to identify the problem; would discuss the problem identified; present the solution; check if the students had other potential solutions; discuss the advantages of that solution and the possible problems; possibly try to apply solutions proposed by students; and discuss upcoming refactorings related to the same artifacts.

P8 would apply the examples in a traditional way: “*I would apply the example in a more traditional way, which is the way I work because I cannot always apply a more active ap-*

<sup>12</sup>[https://api.github.com/search/issues?q=is:pr+%22keyword%22&per\\_page=100](https://api.github.com/search/issues?q=is:pr+%22keyword%22&per_page=100)

<sup>13</sup><https://aserg-ufmg.github.io/why-we-refactor/#/>



#	Question
Q1	How would you employ/use the worked example in any course you teach?
Q2	Why would you use (or not) worked examples?
Q3	How do you think the worked examples can benefit teaching?
Q4	What challenges would you have to include the worked examples in the methodology you already use?
Q5	Would you be able to use these worked examples in a course you already teach?
Q6	Do you use examples in your classes? Are they real examples? Have you used examples from Open Source Software Projects
Q7	What types of topics do you think would be interesting to have worked examples on?

**Table 9.** Interview questions conducted in the first portal evaluation. The questions are just a guide used as a checklist

*proach*". First, they would present the concepts involved in the class, and then they would show the examples—first the problem and then the solution strategy proposed. After that, they would discuss other forms of solution. P2 and P7 would also use it to demonstrate in a practical way the application of concepts in a traditional setting.

Finally, P3 would try to mix the two forms, traditional and active. They would create a slide explaining the problem, another with a wrong "solution", one with the correct solution, and an explanation of the problem's correction. They would also develop an activity for students to try to identify the problem. In this way, one can see the versatility of the worked examples, which can be traditionally used as an example to demonstrate certain concepts, or even as more active activities, which can be performed by students.

According to the instructors participating in the study, the worked examples can benefit teaching by bringing real examples, making things more concrete, leaving the abstract, and providing contact with real projects, facilitating understanding. In the words of P7, "*you can see real situations, especially in Software Engineering where the students have no experience with real projects yet, they think we are talking about very abstract things. As they see a real project, they say, look, this is a real project, people use it here in practice, so it's much more concrete...easier for the student to understand*". P7 added, saying that they already faced situations where they needed a real example, and it was difficult to find.

P3 believes that the examples can help students to interact more in class, even motivating the instructor, as it is a different material, and that it addresses real experiences. P0 also highlights that the examples could help instructors to always be changing, not repeating the same examples, being a way to bring new things to the classes, and always keep up to date

Regarding the difficulties in using the worked examples, the instructors did not point out major challenges, only the reorganization of lectures, but that would not be a big problem considering the benefits obtained. For P2 "*Thinking about the methodology I traditionally use, the challenge is more to reorganize the lectures that are already created, thinking about the material I would have available because the examples I had been using were from traditional Software Engineering books, so always the same example, the biggest challenge is the question of time management, how much time will I have to adapt the material that already exists [...]*".

P3 points out that they would also have no problems using the examples, taking into account the methodology they traditionally use. However, due to the COVID-19 pandemic, P3 has been working remotely and believes that in distance learning education it could be a little more difficult to use such examples. On the other hand, P3 also points out that they

would have to use the examples to conclude, as "*bringing something different can also help students to interact more*", with the examples being a way to motivate students in the virtual environment.

One of the topics pointed out by instructors (P0, P2, P8) as interesting for creating worked examples was "Test", given the fundamentally practical nature of this topic. P8 also pointed out topics related to quality and tools to measure size and complexity. P0 indicated design standards and code reviews, and P7 also cited design standards and *code smells*. P3 would very much like examples related to software modeling and requirements specification; P2 indicated good principles of software programming and maintenance.

After analyzing all the questions, and implementing the improvements in the portal and the cataloged examples, the next step was the execution of another evaluation cycle, conducted again with Software Engineering instructors.

### 7.3 Cycle 5 - Verbal lesson planning by SE instructors

In this cycle, we, once again, invited instructors to use the portal. During a videoconference session, we asked them to think about a course they teach and select one or more examples of refactoring. Then we asked them to verbally plan a class in which they could use the examples. The goal of this assessment was to investigate the usefulness of the examples, what criteria instructors would use to choose the examples on the portal, and how they could use them in a lecture. We also investigated the portal's potential as a teaching resource. Ten Software Engineering instructors participated in this evaluation, according to Table 10, with P1, P3, and P9 participating in previous iterations.

Instructor	University	SE Teaching	Use of OSS
P0	PUC-Rio	15 years	Research
P2	UTFPR	14 years	Research/ contribution
P4	UTFPR	19 years	Research/ contribution
P9	UFMS	2 years	Research/ contribution
P10	UNESPAR	17 years	Research
P11	UFU	20 years	Research
P12	USP	22 years	Research/ contribution
P13	UNIT	22 years	Research/ contribution
P14	UENP	15 years	Research/ contribution
P15	UTFPR	2 years	Research/ contribution

**Table 10.** Information on participants in cycle 5 studies.

In this cycle, we use the think-aloud protocol again. The instructors were invited to narrate their decisions and reflections during the process of searching for the worked examples and preparing the class. We carry out the evaluations individually with each instructor. In this evaluation, the in-

structors did not need to create any material, the preparation of the class was only verbal. First, instructors chose a course and the topic of the course on which they could apply the refactoring examples. With that in mind, they started searching for worked examples that they considered suitable for use in class.

After selecting the worked examples, the instructors carried out the lesson planning, explaining which methodology they would use and where they would apply the selected worked examples. With the class prepared, we asked instructors to explain the criteria used to choose the worked examples. Then, we conducted a post-study interview to understand the portal's potential as a teaching resource. The questions addressed in the interview are presented in Table 11.

In addition, we applied a questionnaire following the Technology Acceptance Model (TAM) (Davis, 1989), to assess the ease of use and usefulness of the portal as a tool. TAM is based on two constructs: perceived utility and perceived ease of use. Perceived utility refers to the degree to which a person believes that using the system can improve their performance, and perceived ease of use refers to the degree to which a person believes that using the system will be effortless (Davis, 1989). The items under evaluation followed a 7-point Likert scale—from (1) Strongly Disagree to (7) Strongly Agree—which is commonly used in TAM questionnaires. The questions TAM questionnaire items in our study are presented in Table 12.

The data collected in the think-aloud and follow-up interview was qualitatively analyzed by the research team, following an open coding approach followed by a thematic analysis to create higher-level categories. We graphically analyzed the responses to the TAM questionnaire.

## 7.4 Cycle 5 - Results - Use of the Portal

The qualitative analysis of the interviews and think-aloud revealed three categories of feedback, namely: **search and navigation**, **difficulties with open source software**, and **use in the future**. Each of these topics will be discussed below.

### Search and Navigation

The topic of **search and navigation** was divided into positive aspects and negative aspects. As a positive aspect, we evidenced simple use and the ease of finding examples. In the words of P14 *“It was a simple, few menus, the contribution part is clear, and the examples too. When you start browsing you can see that all the examples have the same structure, very simple”*. In addition to simplicity, another point discussed was the different ways to search, and simplicity in presenting results. For P0 *“the portal has the advantage of being very simple, and has different ways of searching”*, P13 also points out that *“the search is very simple, returns results in a simple way”*.

Another point mentioned was the intelligibility of topic names of the knowledge areas and the examples. P15 pointed out that *“the names are good, intelligible for what is proposed and the explanation is good and easy to understand”*. In addition, the defined taxonomy for navigating by knowledge areas was pointed out as a positive point, being well organized, in the words of P13 *“the taxonomy used in navi-*

*gating by area is well done, well defined”*. Another positive point presented by P13 was the generate PDF option available for all examples on the portal, this option allows instructors to save the examples, analyze them later, or even send them to students.

The search and navigation negative points were divided customization and general presentation. P9, P12, and P0 felt the need to mark or favorite examples so that one can save and return to a selected example again. In the words of P0 *“it would be interesting to write down the chosen examples, and then go back and log into the site and have it stored”*.

Another point presented by P13 was the possibility of creating a **section to leave feedback** (along with the example evaluation section) about your experience using the example to help other instructors in the process, pointing out what worked and what could go wrong. P12 also pointed out the lack of a section for new examples, that is, a section where the latest examples inserted in the portal would be presented, so that instructors could always keep up-to-date, having access to the latest examples added.

In **general presentation** some suggestions are about the presentation of information on the portal. P10 and P12 mentioned that the examples returned in the search are not sorted and that finding a way to sort them would be interesting. In addition, P2 observed that the portal had no pagination, that is, there was no limitation of results per page, and in some searches, it returned a lot of information in just one page, and that this could make the visualization a little confusing.

### Difficulties With Open Source Software

The **difficulty in using open source projects** was one of the topics addressed by instructors, mainly due to the lack of experience with such projects. Understanding and adapting to open source software was one of the difficulties in using the portal and the examples. According to P12, *“the fact that the portal links to GitHub made it a little difficult for me, but because of my lack of experience”*. P14 also points out that *“a restriction is the GitHub issue, it is nice to have access to the project, but navigating inside it can be a little tricky”*. P12 suggested that *“would be nice if the portal had something to guide users who have no experience with GitHub, a kind of help or tutorial”*. P11 lacked something more “chewed” about what happens on GitHub to reduce the cost of understanding the examples.

### Use In The Future

Question Q8 of *debriefing* addressed **use of the portal in the future**, when it would be possibly already populated for various topics. All instructors participating in the study showed interest in using the portal in the future, pointing out aspects that would attract them. P4, P10, and P13 pointed to the ease of finding good examples and keeping them up to date. When asked why they would use the portal, P10 said that *“one of the aspects that would motivate them to do this would be the ease of keeping the examples up to date. There are several similar examples, and each semester I could use different examples, if there are students who failed the subjects they may have contact with new examples”*.

The aspect that would attract P9 and P11 to use the portal is the fact that it saves time and effort in finding examples.

ID	Question
Q1	How was your experience using the portal?
Q2	What are the challenges in inserting the examples in your lesson plan?
Q3	How do you imagine that using these examples contributes to providing contact with real projects?
Q4	How would students learn from the selected examples?
Q5	How would such examples help with motivation and interest?
Q6	How is it possible to plan a lecture using the portal?
Q7	If you didn't have the portal, how would you look for real examples to use in your classes?
Q8	If the portal was complete, would you use it for all your classes? Why?

**Table 11.** Interview questions conducted in the second portal evaluation.

Perceived Ease of Use	
Q1	It was easy to learn how to use the Portal
Q2	I find it easy to get the Portal to do what I want it to do
Q3	I understood what happened in my interaction with the Portal
Q4	It was easy to gain skill in using the Portal
Q5	It's easy to remember how to use the Portal to find examples for my class
Q6	I find the Portal easy to use
Perceived Usefulness	
Q7	The portal allowed me to search for examples for my class faster
Q8	Using the portal improved my performance in preparing my classes
Q9	Using the Portal improved my productivity in preparing my classes
Q10	Using the Portal made it easy to find examples for my class
Q11	I find the Portal useful for preparing my classes
Q12	Assuming that the Portal is populated for various topics of Software Engineering, I predict that I will use it in the future

**Table 12.** Questionnaire based on TAM related to the Portal of worked examples.

In the words of P9 *“I would use the portal for all courses, mainly due to the reduction of effort”*.

Another aspect that caught the attention of P10 and P2 was the diversity of examples, designs, and languages. In the words of P10 *“the possibility of examples in various languages, since there are courses that can address other languages such as python, ruby [...]”*. For P2 *“the diversity of examples, projects, and languages would be aspects that would make using the portal, and make it a useful and attractive tool”*.

P14, P15, and P0 would use the portal because they knew they would find well-structured examples. For P15 *“examples would be the main aspect that would attract me to the portal, the context and how they are explained, I think it helps too”*.

## 7.5 Cycle 5 - Results - Use of Worked Examples

The results related to the use of worked examples were split into four categories, namely: teaching method, criteria for choosing the examples, benefits of use, and difficulties in using them. The category **Teaching method** describes how instructors thought of using the examples in their classes, being divided into two ways, active and traditional. The category **criteria for choosing examples** addresses which criteria instructors would use when selecting an example to use in classes. In **benefits of use** and **difficulties in use** are presented, respectively, in the instructors' view, the benefits that the use of worked examples could bring to teaching, and the possible difficulties they might have to use such examples.

### Teaching Method

Instructors participating in the study were invited to prepare a lecture in which they would use the examples, after analyzing the information, it was noted that these lectures had

two types of approaches to applying the examples: traditional learning and active learning. In the traditional approach, most instructors would only add examples to existing classes to demonstrate and reinforce the concepts explained in a real way. For example, P12 would go directly to the example within the project, *“play the entire screen with the commit of the example to the students on the projector, and it would show the differences between the original and the refactored program, and it would show and doing the comparisons”*.

Regarding the active approach, P2 mentioned that they could use the examples for an inverted classroom, where they would pass the problem addressed in the example, and let the students discuss it, they could also use the example as an activity. P9 and P13 would use the examples as activities, select some examples that address problems with similar complexities, and pass the problems on to the students to propose solutions. They would later show the solutions given in the examples and compare them with those proposed by the students (it *“would encourage students to discuss and debate the solutions of their colleagues to motivate interaction between them”* –P9).

### Criteria for Choosing Examples

Regarding the criteria used to choose the examples, the instructors mentioned different relevant aspects when making the choice. One of those criteria was the popularity of the project, in the words of P2 *“if I take an example of a well-known project, it might motivate them even more”*. P13 also mentioned the use of projects that students will hear about in the market, as this can arouse interest and motivate them. The project domain, the classes involved and good documentation were criteria cited by P0, a smaller number of classes involved and good project documentation can facilitate the understanding of the example, reducing the cost of understanding for the instructor. P0 also mentioned the specific focus on the problem as a criterion, having an example has a

single focus, and that the project is not mixed with other interests, would facilitate the visualization of the problem and how it occurs, not being necessary understand other code or design elements that may be intertwined with the example.

For P14 the level of abstraction would be the main criterion for choosing or not an example if it would have a model or diagram to help them understand the problem and the context of the example, or if it would just be code: *“getting just the code, didactically, is complicated”*. Another point mentioned by P14, P4, P12, P13, and P8, was the complexity of the example and the technology used, all of which would opt for a simpler example, which would facilitate the understanding. In the words of P12 *“I would look for simpler examples, as the more complex ones would demand more from the students”*.

The course bias and the degree of maturity of the students were also criteria used to choose the examples. P9 would choose the examples taking into account the bias of the course in which they would be applied, and P10, together with the course bias, would also consider the degree of maturity of the students, looking for examples with simpler contexts, which students would find it easier to understand, taking into account the degree of maturity of the class.

P11 would take into account the motivation of the example when selecting it, why that problem is occurring, what led to the presented resolution, and what are the possible implications of the modifications in the project. P2 would choose example by title, being a choice for convenience, not necessarily following a criterion, just thinking about what might be interesting for the class.

### Benefits of Use

All instructors participating in the study believe that the worked examples extracted from OSS projects have potential to bring real experiences to students, as they are examples based on real projects. P13 also points out that *“in addition to providing real contact, it can influence the student to contribute to this type of project as well”*, which can be a way of introducing students to the world of open source projects.

The worked examples can be a way to support teaching, bringing learning to students. However, P10 pointed out an important aspect: *“I am not sure if the students can learn from the examples, it’s a bit strong, but the worked examples are one more tool to show the students that what the instructor is showing us something important”*.

Still, in general, all instructors agreed that the worked examples can impact the motivation and interest of students, mainly because it is a practical material, in which students can actually see the application of concepts and techniques in projects of real software. P2 also adds that the design choice can be a factor that influences these aspects, in their words, the use of worked examples: *“it can motivate for being a practical example, and there are other aspects too if I take an example of a well-known project, can motivate them even more”*.

Another aspect that can influence the motivation and interest of students is the active nature with which examples can be used, P10 points out that *“The use of examples in class can help in the motivation because the examples can be used to do more dynamic activities, such as giving the*

*problem and asking students to propose solutions and then comparing the solutions they make with the solution in the example”*. Performing more dynamic activities can engage students more and motivate them in the learning process.

### Difficulties in Use

One of the difficulties pointed out was reorganizing the lectures. In the words of P11, *“to use the examples I would have to modify the methodology a little because I don’t work with real examples...I would have to rearrange the lectures to insert the worked examples”*. It is important to point out that even in the face of such difficulty, P11 mentioned that despite having to tinker with the methodology, the cost would be worth it, given the benefits it would have later.

Another difficulty pointed out was the technologies used. The examples of projects that approach technologies that are different from those commonly used in the classroom may require more time for application. P14 pointed out that *“when we enter an application level, maybe we don’t work on the same platform, [...], and maybe I wouldn’t be addressing this in my course, because of the time, which is not possible to work with several technologies”*. Other difficulties mentioned by P14 were the cost of understanding the worked examples, and the difficulty of inserting something that is not the instructor’s authorship in the classes. In P14’s words, *“the cost of understanding the example to adapt the lesson, the difficulty of having to insert something external in the content, such as an example created by others”*. P12 also pointed out that the insertion of new material can bring some difficulties, but that it is part of class preparation.

Some instructors mentioned that they would not have difficulties applying the examples due to the methodology they already use. Some small difficulties could be related to the context where the example would be used, or how it would be used, these restrictions being related to the course or the degree of maturity of the students. P2 for example, would use the examples for a course on software analysis and design, so I would have to make some adaptations to the example, but because of the context in which it would be working, this would not necessarily be a difficulty related to the use of the example, but rather to the course that would make the application, in their words, *“In the context of the course, the difficulty would be to understand a little more of the project, due to what the course asks for, to understand the connection and hierarchy of the classes, to generate the UML”*.

## 7.6 Cycle 5 - Results - TAM Questionnaire

The questionnaire applied to instructors participating in the second evaluation of the portal contained 12 questions, according to 12, based on the TAM, with answers on a 7-point Likert scale. In this questionnaire, an open question was also presented for instructors to leave comments if deemed necessary. In 12 we present the results of the items present in the questionnaire.

In general, instructors who participated in the study agreed that it was easy to learn to use the portal and that it was easy to get the portal to do what they expected. In addition, they also agreed that they were able to understand what happened during the interaction with the portal and that it was easy to

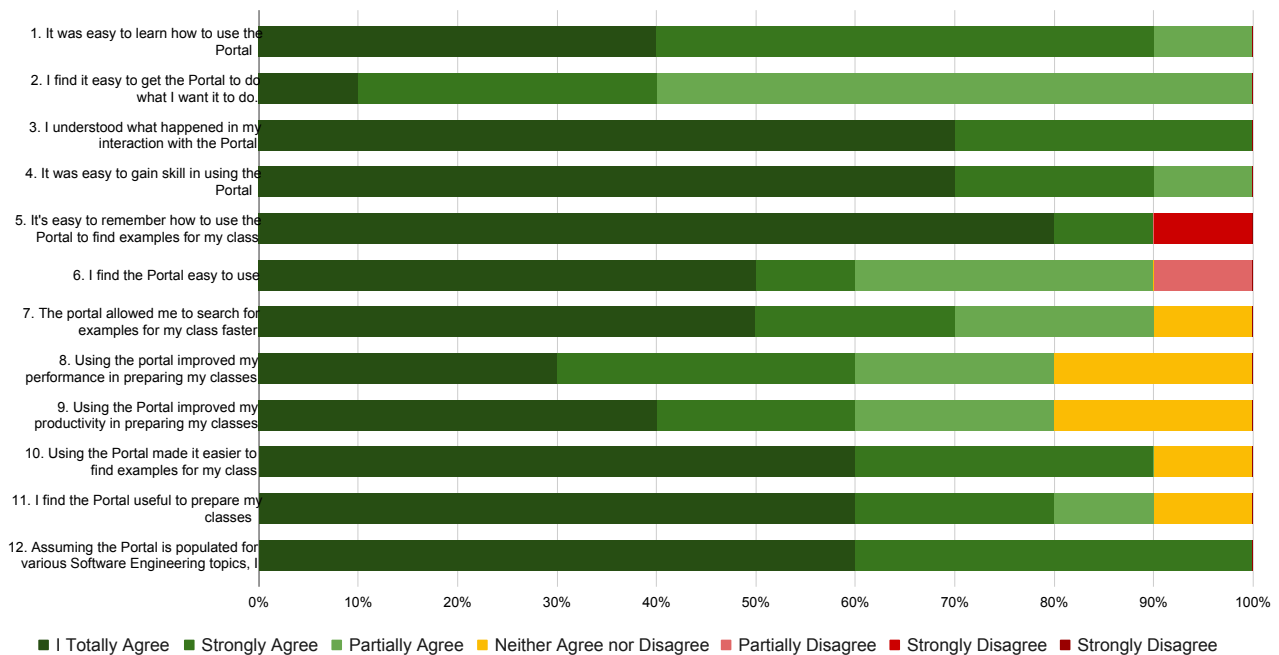


Figure 12. Results obtained in the application of the questionnaire based on the TAM.

gain skills in using it.

Figure 12 shows a clear trend of agreement, with only two scores showing disagreement (one on item 5, and the other on item 6). Both of the negative answers came from the same instructor.

Only 3 instructors out of 10 answered the optional open question left at the end of the questionnaire. One of them suggested that the layout could be improved to focus on usability, and another suggested that the example could contain some color marking to demonstrate the level of complexity and group examples of similar projects. In summary, the TAM results indicate that the portal is promising and could be well-received by Software Engineering instructors.

## 7.7 Cycle 6 - Complete lesson planning by SE instructors

In this cycle, the instructors used the portal as a support for the complete preparation of a lecture. This preparation involved creating a lesson plan and the materials that would be used in class, such as slides, activities, and complementary materials. The topic selected for this evaluation was refactoring again, and it was up to the instructors to prepare a lecture that addressed the selected topic. To conduct this study, each instructor joined the researchers in two meetings that lasted around 50 minutes each. They had to carry out some activities between the meetings.

The focus of this cycle was to collect the instructors' feedback about preparing a lecture using the portal as support material, and their perceptions about the use of worked examples in the classroom. Information about the instructors who took part in this assessment cycle is presented in Table 13. The study was conducted individually with each instructor.

In the first meeting, we explained the study protocol and the instructor consented to participate. Then, we present the

Instructor	University	SE Teaching	Use of OSS
P2	UTFPR	14 years	Research/ contribution
P3	UNESPAR	5 years	Research
P9	UFMS	2 years	Research/ contribution
P16	UFABC	3 years	Use Only
P17	USP	15 years	Research/ contribution
P18	UFBA	7 years	Research/ contribution

Table 13. Information on participants in cycle 5 studies.

portal, making a shared folder available with a bulletin board in which the instructors could leave feedback on the positive and negative aspects of using the portal throughout the study. We also provided a lesson plan template so the instructors could use a guideline. After presenting the portal, the instructor chose a course and started the lesson plan, detailing the methodology and how they would use the examples. At the end of the first meeting, we gave instructions about what was expected for the next meeting (presentations, activities, complementary materials).

The first meeting resulted in a first version of the lesson plan, in which the instructor would make use of the portal examples in some way. This version of the plan was open so that the instructors could improve as they saw fit. The second meeting was scheduled according to the time established by the instructors so that it was possible to prepare the necessary materials for the class.

The second meeting was recorded, with the consent of the participants. At this meeting, we invited the instructors to present their lesson plan and the materials developed. We then discussed the positive and negative aspects of using the portal. Finally, we conducted a post-study interview to collect feedback on the process of preparing the lectures aided by the portal. The questions asked in the interview are available in Table 14.

RQ	Question
Q1	Could you summarize your experience using the portal?
Q2	What benefits and challenges can you see in the application of the worked examples, related to learning?
Q3	What were the difficulties in inserting the worked examples in the lesson plan?
Q4	Would these difficulties prevent you from using the worked examples in your classes? Why?
Q5	In the lecture you prepared, are there factors that could cause the application of the worked examples to not have the expected result?
Q6	What is the role of the portal in the preparation of the lesson using the worked examples?
Q7	When preparing the class, did you change the methodology you use? If so, why?
Q8	Do you intend to conduct the planned lecture?

Table 14. Interview questions used in Cycle 6 post-study

## 7.8 Cycle 6 - Results

The 5 study participants showed different ways of using the worked examples retrieved from the portal in different courses. Some instructors planned active approaches, such as creating activities and assignments, and others as a complement to the lectures they already use to teach. P16 prepared a class for the Object Oriented Programming course and planned to use the examples as an activity, which would be automatically corrected by the Moodle virtual environment, something they already used in class.

P9 also planned to use the examples as an assignment in the Software Measurement course. The instructor pre-selected some worked examples and would let the teams choose the examples and projects they would like to work with. The purpose of P9 was for students to explore the project from which the example was extracted and apply software measures. P18 would use the examples in a Software Engineering course, in a class on Design Patterns, to show the before and after the code implementation, making an association of improvements with design patterns.

P2 and P3 planned an expository lecture, in which they added examples. P3 prepared a lecture on refactoring for the Evolutionary Software Maintenance course, which was supported by a slideshow, in which they incorporated the examples among the concept explanations, to show the real application of what they were presenting. P3 also pointed out that they could apply an activity using the examples. P2 adapted one an Object-Oriented Analysis and Design course lecture which focuses on good *design* practices. They added the examples at the end of the lecture, to show refactoring as a good practice and take real examples of *design* solutions for students. Lastly, P17 planned to use the examples in two ways, first as P2 and P3, during the presentation of the content to demonstrate the application of the concepts, and then use it as an activity to try to put the presented content into practice.

### 7.8.1 Positive and negative feedback

A document was also made available so that each instructor could leave their *feedback* on the positive and negative aspects of using the portal. One of the **positive aspects** mentioned was the facility to find practical and real examples explaining the context in which they are inserted. P3 wrote that *“the portal helps me by providing lots of practical and real-world examples of refactoring. This is pretty cool to have, as most of the examples we find in literature or book materials are old or too abstract for students to understand”*. For P17 *“the portal helped with the selection of examples for the lesson and for composing exercises”*. Therefore, we could

evidence that the portal facilitates the search for examples to explain concepts discussed in class, and even to generate practical exercises.

Also on the positive side, P9 mentioned that *“scenarios help to create variations to the examples given in the classroom”*. P18 pointed out that *“the examples I use in class end up being repeated semester after semester, and are not always “up to date”, with the most modern practices of Software Engineering. So, I think that a portal with examples of more current and robust systems is very useful for the development of teaching material.”*

Another positive aspect mentioned by the instructors was the possibility of having examples for different programming languages, P3 pointed out that *“I was able to choose the type of programming language I would like to present the examples with”*. P9 also points out that *“the variety of languages helps strengthen the abstraction aspect”*. In other words, the fact that the portal enables the cataloging of examples in different types of languages was seen as a positive aspect for the instructors participating in the study.

According to P9, *“the categories helped me to be assertive in the example filter step”*, that is, the categories defined in the portal structure helped the instructor to more easily find the examples they wanted. P16 also pointed out that the portal helped them to define topics and what they could spend in class, in addition to finding the format of the examples as a facilitator in the construction of slides, due to the established format and the elements present in the examples.

Regarding the **negative aspects**, one of the factors mentioned by the instructors was the lack of a level of complexity/difficulty of the examples, or of metrics to know how big or complex the project addressed in the example is. P2 wrote that *“as the students are from the third semester, some examples involved many classes and lines of code. Not being good examples for students. Having some information about the example (APIs involved, number of lines added/removed, number of classes involved) could help me filter the information”*. P9 cites the level of difficulty and comprehension of the examples, *“scenarios could be classified about the level of difficulty, comprehension”*, and also complements talking about metrics *“I missed some metrics like (lines of code, at least), right on the first project selection page so I can make my choice easier”*.

P2 indicated the difficulty of searching through the search bar: *“when I performed some queries, the terms came back “without any record”. I had to browse the list of cases, enter some examples until I chose the best case”*, and pointed out that this problem could be because the portal still has few examples, being populated only for one topic. For P9 *“Some*

projects are not well documented, which makes it difficult for some groups of students to understand”, in addition, P9 was in doubt whether the projects had been curated, in terms of quality, and even the same applicability, missing some metrics to assess the use of projects in the classroom, in addition to the comments and stars scheme.

### 7.8.2 Post-study interview

Regarding the feedback obtained in the interview, in general, the instructors viewed the experience as something positive. P16, who is already accustomed to using examples, found that the portal helped him in searching for material. P3 highlighted that the worked examples available on the portal provide a context that common examples usually lack since “[...]most books do not contextualize the examples so well, and when using real examples, they are inserted within a real context, and this is very important to use”.

The instructors also mentioned the convenience that the portal provides. According to P2, “it’s better to search in a place where things are more contextualized than to go mining projects on GitHub”. P9 also emphasized the idea that the portal makes it easier to search for examples, as searching on GitHub can be challenging. For him, “It’s clear that [the portal] helps a lot in structuring thinking and streamlining because GitHub doesn’t have these categories”. P17 also stated that the experience of using the portal was “very smooth”.

Another point mentioned about the experience with the use of the portal was the fact that it could offer something different to the students, providing new perspectives. For P2, “if you stick to using only examples from books, they tend to be more didactic, but you get tired because you keep working on the same examples, so it’s good to present other perspectives to the students, and if it’s part of something practical, an open-source project, it’s interesting to show”.

P9 mentioned that they liked using the portal but confessed that they felt a lack of balance between good examples and bad examples, in other words, they missed incorrectly worked examples. For them, “A good Software Engineering instructor, regardless of the course, will not only show wonderful projects, but they also have to show the chaotic”.

Regarding the benefits of using worked examples, most instructors focused on the practical and real-world aspects of the examples, which allows students to see real situations. For P3, having contact with the real application of concepts can help students mature both as professionals and in terms of learning. P2, P9, P17, and P18 also believe that real-world experiences in the classroom can contribute to learning. In P18’s words, “[...] software engineering subjects have to be practical; the theoretical part is super important, but if there’s no practical application of it, knowledge may not be easily retained by students”. And the portal can be a tool to bring that practicality in some way.

As for the challenges, for P3, applying worked examples requires a good explanation of the context so that students can understand all the nuances; otherwise, the examples may not have the expected effect. According to P9, using a highly detailed example can hinder students from developing critical and analytical skills. Additionally, selecting appropriate

examples for the course was mentioned as a challenge for instructors, as not every software project may be understandable to students. According to P3, to facilitate learning, examples should be chosen in a language that students know and with content that has already been covered. P2 also mentioned the students’ level of maturity, stating that selecting and adequately explaining the example can bring benefits to learning.

When asked about the challenges of integrating examples into their lesson plans, P17 and P2 stated that they had no difficulties because they already worked with examples, so it was easy to incorporate them into their classes. In P2’s words, “the integration was smooth; I basically incorporated the example, and I could even print the examples to give to students as supplementary material”. For P16 and P9, the main challenge was understanding the complexity of the examples and analyzing what could be considered easy or difficult for the students. For P3, the only difficulty was deciding what to present and which parts of the worked example to use, but after doing this the first time, they did not face any more difficulties. P18 highlighted that using examples associated with the topics addressed in the class was a challenge, especially when there were few examples from the same project.

However, all the instructors believe that the benefits the portal can provide outweigh the few challenges. For them, the role of the portal would be to assist in class preparation, facilitating the search for practical examples, project selection, supplementary material, and even materials for practical assignments.

Most instructors mentioned that they did not change the methodology they use to integrate examples; only P9 said that if it weren’t for the portal, they would probably make some changes in how the activity is applied. Most of them are already accustomed to working with examples, so they simply added them to the materials they already use. In P16’s words, “I use a lot of examples in the classroom, and the vast majority of the examples I use to present concepts I get from websites, from GitHub itself, I also use real examples, so I didn’t see any problems in including these examples in the process”. P2 stated that they did not change, but they would not see any problem with having to do so. In their words, “what I did was try to reuse the content I already had and incorporate it into a lesson I had already prepared, but I don’t see why not use the portal to change the approach, try to do something more active”.

To conclude the interview, the instructors were asked if they would implement the planning they had discussed, and all of them said yes. However, the studies were conducted at the end of the semester, so none of the instructors had the opportunity to implement the plan at that time.

## 7.9 Results and Discussion - Stage 2

As the cycles progressed, we improved the portal, included new elements in the example structure, navigation was improved, and the search and presentation of information was improved. These improvements were necessary to meet the requirements established for the development of the portal. We also obtained answers about the usefulness of the portal as a support in the teaching of Software Engineering.

### How was the overall perception about using the portal and the examples?

All the instructors who participated in the studies found ways to fit the worked examples in their lectures. Some in a more traditional way, just including the examples in the classes they are already used to teach. Others have found more active ways to apply the examples, either through an activity, through classroom discussions, or even in flipped classrooms. Thus, it is possible to observe that the way of structuring the worked examples allows versatility. In this way, the examples can be used in different ways, addressing different methodologies, whether traditional or active. The purpose of **requirement 3** was for the portal to guide instructors in applying the worked examples. Thus, the versatility of the examples will allow the instructor to adapt the material to its context without many difficulties, which can facilitate the application.

When asked why they would use the worked examples, the instructors mentioned that it would be a way to take real examples to the classroom, since, in most cases, it is difficult to find this type of material. Another point addressed was that the portal would be a way of always updating the examples, bringing new examples to students each semester, in addition to bringing differentiated material. The fact that the examples provide contact with something real, was seen as something beneficial for teaching, as it makes things more concrete for students, in addition to providing contact with real projects. Such information is in line with the objective of **requirement 2**, which says that the portal should openly provide real examples to support the teaching of Software Engineering, reducing instructors' time to search for materials and resources.

About the benefits of using the examples, the instructors mentioned that through them students can have contact with projects, environments, and real problems of software development, being a way to show that what the instructor is explaining is important and has relevance in a real environment. In addition, this type of material can act on the motivation and interest of students, help them to mature professionally and in terms of learning, and even influence them to contribute to OSS projects. Most instructors are already used to working with examples, so the portal would be a way to help in the search and update of examples.

Regarding the difficulties that instructors could have to use the examples, the main one would be the reorganization of the classes, to insert the examples. Another difficulty could be the technology used in the projects, which may not coincide with those taught in the courses. In addition, the cost of understanding the examples was also a factor cited, given that very complex examples may require a little more study time for the instructor to take to the classroom. The insertion of something new in the class, which is not authored by the instructor, can also be a factor of difficulty.

In the search and navigation aspects, positive and negative aspects related to the usability of the portal were presented. The instructors pointed out positive elements that facilitated their use and gave tips on new features that could be useful in the future, in addition to suggesting changes in the presentation of some elements. Such suggestions will be investigated in future works, which will be focused exclusively on the us-

ability of the portal.

Another focus was the difficulty that some instructors pointed out in the use of OSS projects, especially those who had no experience with this type of project. They suggested a tutorial on the portal to help people unfamiliar with GitHub, or open source projects, and to provide more contextualized examples of what happens in the projects. In addition to usability issues, these suggestions will also be investigated in future work. The focus of this work was to investigate the usefulness of the portal as a support tool in the teaching of Software Engineering.

### Diverse criteria to choose appropriate examples

The criteria for choosing the examples to use in class were diverse. Some instructors prioritized the popularity of the project, domain, classes involved, and whether the project had good documentation. Others mentioned the level of abstraction of the example if, in addition to the code, there was some kind of model to guide. The context of the example was another criterion adopted. If the context is easy to understand, and the appropriate complexity takes into account the bias of the course and the degree of maturity of the students. To summarize, the choice of the example varied a lot according to the course in which it would be used, the level of maturity of the students to which the example would be applied, and the profile of the instructor.

One of the points to be observed when choosing the example was their complexity. The complexity must be following the level of maturity of the class in which the example will be applied. Using very complex examples in non-advanced classes can confuse students and misunderstand the concepts applied to the example.

Another aspect that should be given attention is related to the content covered in the examples and the technologies used. It is important to select examples with content that students already have sufficient prior knowledge to understand the problem addressed, and the proposed solution. The technologies used in the examples should preferably be the same as those used in class, or at least with a functioning similar to something that students already know. In this way, the cost of understanding will be lower, as students will have contact with something already familiar to them.

The popularity of the project can also be an interesting point to be analyzed. Choosing examples of projects that students know can be a motivating factor, knowing that that example was taken from a real project, that they are aware of, and may even have already used it.

### A good perspective for the future.

Regarding the use of the portal in the future, all instructors stated that they would use it, for several reasons. The portal provides the facility to find real examples and keep them updated, saving time and effort in the search. In addition, it has a diversity of examples and projects, in different programming languages, and provides practicality to show real, well-structured examples. In this way, the portal can be an important support material for teaching Software Engineering, given the benefits it can bring to the instructor.

Regarding the application of the examples, they proved to be very versatile during the execution of the studies. In



this way, the instructor will be able to use it as they see fit, either to exemplify concepts, create activities, conduct group discussions, or even for an inverted classroom. It is up to the instructor to analyze the course and the class in which they will apply and to identify the best way to take the examples.

Based on the results obtained in the studies, it was possible to create **guidelines to assist instructors in adopting the worked examples in the teaching of Software Engineering**. Such examples are potentially beneficial for teaching. However, some care is needed when selecting the examples and applying them in the classroom.

## 8 Rigor Cycle

In DSR, the Rigor Cycle is related to the knowledge base and research contributions, since the DSR is not only linked to the processes of development of the artifact, it is also necessary to evolve the knowledge base of the researched area. The rigor of the research guarantees its credibility, honesty, and trust, such rigor is guaranteed when the researcher follows methods already known, used, and validated, and preferably widely accepted methods (Hevner and Chatterjee, 2010).

Thus, to achieve the expected rigor for the research and generate relevant contributions, we carry out studies on the existing methods, selecting the most appropriate methods for each stage of the studies. During the execution of the DSR cycles, we used different methods, such as focus groups, think-aloud verbal protocol, interviews, and questionnaires, and the information obtained was analyzed qualitatively.

One of the main contributions of the research is the creation of a portal for cataloging the worked examples extracted from open source software projects. The portal can assist instructors in the process of bringing OSS projects into the classroom, providing students with contact with real projects, and assisting in the search for real examples. In addition, the portal is an effort to disseminate the use of worked examples in the teaching of Software Engineering.

The adoption of worked examples extracted from open source software projects can contribute, as a side effect, to the strengthening of OSS communities, since students, and even instructors, can feel motivated to contribute to these projects, by having closer contact with this environment.

In addition, we can cite other contributions, such as:

- Identification of the difficulties faced by instructors in adopting open source software projects in the educational environment;
- Creation of guidelines to guide instructors in the adoption of worked examples in Software Engineering courses;
- Creation of a template to create and structure worked examples extracted from open source software projects;
- Analysis that shows that the worked example portal can reduce the challenges faced by instructors in the search for real examples;

As a complement to this work, a practical application of the worked examples created and evaluated during the studies described in our previous work Tonhão et al. (2021). Such work presents the students' perspective on the use of worked

examples in SE disciplines since the focus of this work was only on the instructor.

## 9 Limitations and Threats

Like all scientific research, this work has some limitations and threats. We summarize those that offer threats to our results and conclusions.

**Sampling Limitations:** First, although instructors with different profiles were selected to participate in the studies, the sample does not represent all instructors of higher education. Therefore, it is not possible to guarantee that all perceptions about the portal have been analyzed and that all challenges and advantages of use have been identified. The discussions carried out in this work are based only on the information provided by the instructors of higher education participating in the studies, which may represent only a portion of this group.

**Bias in Responses:** In addition, instructors' comments and responses during studies using focus group, think-aloud, or interviews, run the risk of being influenced by a confirmation bias, in which study participants can provide information in a meaningful way. biased. To mitigate such a potential problem, in all studies, instructors were encouraged to provide their comments and opinions sincerely, with the researcher making it clear that they were looking for problems in the approach.

**Questionnaire Design:** The interviews were composed of open-ended questions related to the usefulness of the portal and the use of examples. Such questions were intended to collect instructors' perceptions about certain points. To validate the questions, they were all discussed in meetings with two other more experienced researchers. The objective was to verify that the questions met the intended objective, were clear and objective, and that the script was adequate. Additionally, in the survey, a question that was not included, and may be important, relates to the area of Software Engineering in which the worked examples are applied by the instructors.

**Subjectivity in Data Analysis:** The subjectivity of the classification and analysis of the collected data can be another threat to the validity of the results of the work. In an attempt to minimize this threat, an approach was used in which all analyses were performed based on the speeches and comments of the instructors participating in the studies. In addition, we held meetings with two other researchers to discuss the analysis process, to guarantee a better interpretation by mutual agreement.

**Limited Scope of Studies:** All studies conducted with higher education instructors focused on the topic of refactoring, that is, instructors had contact only with examples of this subject to form their opinions. It is not possible to define whether the results obtained would be the same for other SE topics since there are topics that are unlikely to benefit from this approach.

**Portal Language:** A limitation of the research is that the portal, being exclusively in Portuguese, may restrict the participation and access of researchers and users who do not speak this language, thus limiting the generalization of results and the applicability of the tool in international con-

texts.

## 10 Conclusion

Open source software projects can be a great alternative to support teaching in Software Engineering courses and have been used by some instructors. The adoption of these projects can bring several benefits, such as the possibility of working on projects with realistic sizes and complexities, and of developing necessary skills and experiences in the industry, in line with professional software development. In addition, OSS projects can help students to awaken creativity, and increase motivation and understanding of Software Engineering processes.

Instructors looking to adopt these projects in the classroom may face some challenges, such as choosing the appropriate project, difficulties in familiarizing themselves with the environment of open source software projects, and difficulties in deciding how to take these projects into the classroom, causing more time to prepare the classes. These projects can be a possible source for extracting artifacts that can become material for creating potential worked examples, which could facilitate the adoption of software projects in the classroom, and provide contact with real examples. Thus, in this study, we argue that the worked examples extracted from open source projects can be used to exemplify in a real way, the concepts, and techniques of Software Engineering.

The adoption of worked examples is a common practice in different areas and levels of education, due to the different benefits. Some benefits are reduced cognitive load, less learning time, better conceptual understanding, and being able to act directly on students' motivation. In the teaching of Software Engineering, creating examples can be a challenge for instructors, given the effort required to find and structure real examples and to keep up with the rapid changes in the world of software development.

Therefore, in this work we implemented a portal to catalog worked examples extracted from OSS projects. Our goal was to reduce the challenges faced by instructors in adopting OSS projects in education and in the search for examples that demonstrate real applications of Software Engineering concepts or techniques.

The evaluations of the portal showed overall positive feedback, evidencing that the portal can aid instructors in teaching Software Engineering. For the instructor, the portal can act as a facilitator, helping to keep their examples up to date, and in diversification, always bringing new examples to students. In addition, it can help to reduce the time to search for examples (mainly real examples) and, consequently, the time to prepare the lesson.

It is important to emphasize that the worked examples obtained from open source software projects are not the solution to all problems for teaching Software Engineering. Some topics can hardly be benefited from the vision of the worked examples. In addition, the work did not aim to investigate how to attract people to the portal, or how to popularize the portal for as many topics as possible. We make the portal available in an open way, to allow instructors and professionals of Software Engineering to consult and use the cataloged

examples, and in the future contribute to their population.

While conducting studies, instructors gave tips on new features that could enhance the portal, in addition to suggesting changes related to usability. Such suggestions will be investigated in future works, which will be focused exclusively on the usability of the portal, and extra features related to the topic addressed. Furthermore, there are plans to investigate the possibility of implementing a Recommendation System on the portal so that, based on the instructor's needs, the portal can recommend worked examples that best fit the instructor's needs and the disciplines being taught.

The planning and application of the worked examples in the classroom is something that we aim for in the future. The goal is to get feedback from the instructors about the application, since, so far, only the part of the lesson planning has been analyzed on the portal. Student feedback is also expected in the future to better understand the impact that practical examples can have on learning Software Engineering courses since the student perspective was not explored in this study.

## 11 Acknowledgments

The authors would like to thank the financial support from CAPES (Foundation Coordination for the Improvement of Higher Education Personnel) - (Financing Code 001).

## References

- ACM (2015). *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. ACM.
- Akhuseyinoglu, K., Hardt, R., Barria-Pineda, J., Brusilovsky, P., Pollari-Malmi, K., Sirkiä, T., and Malmi, L. (2022). A study of worked examples for sql programming. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1*, pages 82–88.
- Atkinson, R. K., Renkl, A., and Merrill, M. M. (2003). Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *Journal of educational psychology*, 95(4):774.
- Baltes, S., Dumani, L., Treude, C., and Diehl, S. (2018). Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. In *Proceedings of the 15th international conference on mining software repositories*, pages 319–330.
- Bofferding, L., Kocabas, S., Aqazade, M., Haiduc, A.-M., and Chen, L. (2022). The effect of play and worked examples on first and third graders' creating and debugging of programming algorithms. In *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions*, pages 19–29.
- Booth, J. L., McGinn, K. M., Young, L. K., and Barbieri, C. (2015). Simple practice doesn't always make perfect: Evidence from the worked example effect. *Policy Insights from the Behavioral and Brain Sciences*, 2(1):24–32.
- Bourque, P. and Fairley, R. E. (2014). *Guide to the software*

- engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.
- Buchta, J., Petrenko, M., Poshyvanyk, D., and Rajlich, V. (2006). Teaching evolution of open-source projects in software engineering courses. In *2006 22nd IEEE International Conference on Software Maintenance*, pages 136–144. IEEE.
- Carroll, W. M. (1994). Using worked examples as an instructional support in the algebra classroom. *Journal of educational psychology*, 86(3):360.
- Chen, X., Mitrovic, A. T., and Matthews, M. (2019). Learning from worked examples, erroneous examples and problem solving: towards adaptive selection of learning activities. *IEEE Transactions on Learning Technologies*.
- Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, pages 319–340.
- Demeyer, S., Ducasse, S., and Nierstrasz, O. (2008). *Object-oriented reengineering patterns*.
- Deng, L., Dehlinger, J., and Chakraborty, S. (2020). Teaching software testing with free and open source software. In *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 412–418. IEEE.
- Dorodchi, M. and Dehbozorgi, N. (2016). Utilizing open source software in teaching practice-based software engineering courses. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE.
- Ellis, H. J., England, W. N., Morgan, B., Oregon, W., Hislop, G. W., Coleman, B., and Pulimood, S. M. (2015). Software engineering learning in hfoss: A multi-institutional study. *age*, 26:1.
- Ellis, H. J., Morelli, R. A., and Hislop, G. W. (2008). Work in progress—challenges to educating students within the community of open source software for humanity. In *2008 38th Annual Frontiers in Education Conference*, pages S3H–7. IEEE.
- Garces, S., Vieira, C., Ravai, G., and Magana, A. J. (2022). Engaging students in active exploration of programming worked examples. *Education and Information Technology*, pages 1–18.
- Gaweda, A. M., Lynch, C. F., Seamon, N., Silva de Oliveira, G., and Deliwa, A. (2020). Typing exercises as interactive worked examples for deliberate practice in cs courses. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 105–113.
- Gehring, E. F. (2011). From the manager’s perspective: Classroom contributions to open-source projects. In *2011 Frontiers in Education Conference (FIE)*, pages F1E–1. IEEE.
- Hevner, A. and Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems*, pages 9–22. Springer.
- Hevner, A., March, S. T., Park, J., Ram, S., et al. (2004). Design science research in information systems. *MIS quarterly*, 28(1):75–105.
- Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4.
- Holmes, R., Allen, M., and Craig, M. (2018). Dimensions of experientialism for software engineering education. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET ’18*, pages 31–39, New York, NY, USA. ACM.
- Hu, Z., Song, Y., and Gehring, E. F. (2018). Open-source software in class: Students’ common mistakes. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training, ICSE-SEET ’18*, pages 40–48, New York, NY, USA. ACM.
- Jaccheri, L. and Osterlie, T. (2007). Open source software: A source of possibilities for software engineering education and empirical software engineering. In *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS’07: ICSE Workshops 2007)*, pages 5–5. IEEE.
- Jaspers, M. W., Steen, T., Van Den Bos, C., and Geenen, M. (2004). The think aloud method: a guide to user interface design. *International journal of medical informatics*, 73(11-12):781–795.
- Kontio, J., Lehtola, L., and Bragge, J. (2004). Using the focus group method in software engineering: obtaining practitioner and user experiences. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE’04.*, pages 271–280. IEEE.
- Lessa, M. S. B. and von Flach G. Chavez, C. (2020). An approach for selecting floss projects for education. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 463–472.
- McGinn, K. M., Lange, K. E., and Booth, J. L. (2015). A worked example for creating worked examples. *Mathematics Teaching in the Middle School*, 21(1):26–33.
- Metrólho, J., Ribeiro, F., Graça, P., Mourato, A., Figueiredo, D., and Vilarinho, H. (2022). Aligning software engineering teaching strategies and practices with industrial needs. *Computation*, 10(8):129.
- Montagner, I. d. S. and Kurauchi, A. T. N. (2022). Learning professional software development skills by contributing to open source projects. In *2022 IEEE Frontiers in Education Conference (FIE)*, pages 1–7. IEEE.
- Morgan, B. and Jensen, C. (2014). Lessons learned from teaching open source software development. In *IFIP International Conference on Open Source Systems*, pages 133–142. Springer.
- Müller, M., Schindler, C., and Slany, W. (2019). Engaging students in open source: Establishing foss development at a university. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*.
- Nandigam, J., Gudivada, V. N., and Hamou-Lhadj, A. (2008). Learning software engineering principles using open source software. In *2008 38th Annual Frontiers in Education Conference*, pages S3H–18. IEEE.
- Nascimento, D. M., Chavez, C. F., and Bittencourt, R. A. (2018). The adoption of open source projects in engineering education: a real software development experience. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

- Nivelstein, F., Van Gog, T., Van Dijck, G., and Boshuizen, H. P. (2013). The worked example and expertise reversal effect in less structured tasks: Learning to reason about legal cases. *Contemporary Educational Psychology*, 38(2):118–125.
- Papadopoulos, P. M., Stamelos, I. G., and Meiszner, A. (2013). Enhancing software engineering education through open source projects: Four years of students' perspectives. *Education and Information Technologies*, 18(2):381–397.
- Pereira, J. (2021). Leveraging final degree projects for open source software contributions. *Electronics*, 10(10):1181.
- Pereira, J. and Díaz, Ó. (2022). Open-source software in the classroom: Empowering students to self-select projects to contribute. *IEEE Transactions on Education*, 65(4):553–561.
- Pereira, J. and Pitxitxi, C. R. M. (2020). Capstone projects aimed at contributing to consolidated open source projects: a practical experience. *education*, 2:6.
- Pinto, G. H. L., Figueira Filho, F., Steinmacher, I., and Gerosa, M. A. (2017). Training software engineers using open-source software: the professors' perspective. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*, pages 117–121. IEEE.
- Raj, R. K. and Kazemian, F. (2006). Using open source software in computer science courses. In *Proceedings. Frontiers in Education. 36th Annual Conference*, pages 21–26. IEEE.
- Rourke, A. and Sweller, J. (2009). The worked-example effect using ill-defined problems: Learning to recognise designers' styles. *Learning and Instruction*, 19(2):185–199.
- Sadiku, M. N. O., Olasupo, K., and Nelatury, S. R. (2012). What professors do. *IEEE Potentials*, 31(3):10–11.
- Schwonke, R., Renkl, A., Krieg, C., Wittwer, J., Alevén, V., and Salden, R. (2009). The worked-example effect: Not an artefact of lousy control conditions. *Computers in Human Behavior*, 25(2):258–266.
- Silva, D., Tsantalis, N., and Valente, M. T. (2016). Why we refactor? confessions of github contributors. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 858–870.
- Silva, F. G., Brito, M. S., Tavares, J. V. T., and Chavez, C. v. F. G. (2019). Floss in software engineering education: Supporting the instructor in the quest for providing real experience for students. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*, pages 234–243. ACM.
- Silva, F. G., dos Santos, P. E. D., and von Flach G. Chavez, C. (2020a). Do we use floss in software engineering education? mapping the profiles and practices of higher education teachers from brazil. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 473–482.
- Silva, F. G., Lessa, M. S. B., da Luz Lopes, N., and von Flach G. Chavez, C. (2020b). Teaching uml models with floss projects: A study carried out during the period of social isolation imposed by the covid-19 pandemic. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*, pages 483–492.
- Skudder, B. and Luxton-Reilly, A. (2014). Worked examples in computer science. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*, pages 59–64. Australian Computer Society, Inc.
- Smith, T. M., McCartney, R., Gokhale, S. S., and Kaczmarczyk, L. C. (2014). Selecting open source software projects to teach software engineering. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 397–402. ACM.
- Sweller, J., Van Merriënboer, J. J., and Paas, F. G. (1998). Cognitive architecture and instructional design. *Educational psychology review*, 10(3):251–296.
- Tan, S. H., Hu, C., Li, Z., Zhang, X., and Zhou, Y. (2021). Github-oss fixit: Fixing bugs at scale in a software engineering course. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 1–10. IEEE.
- Tonhão, S., Colanzi, T., and Steinmacher, I. (2021). Using real worked examples to aid software engineering teaching. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering*, pages 133–142.
- Tonhão, S. d. F., Colanzi, T. E., and Steinmacher, I. (2020). A portal for cataloging worked examples extracted from open source software. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 493–498.
- Toukiloglou, P. and Xinogalos, S. (2022). Ingame worked examples support as an alternative to textual instructions in serious games about programming. *Journal of Educational Computing Research*, page 07356331211073655.
- Van Gog, T. and Kester, L. (2012). A test of the testing effect: acquiring problem-solving skills from worked examples. *Cognitive Science*, 36(8):1532–1541.
- Van Gog, T., Kester, L., and Paas, F. (2011). Effects of worked examples, example-problem, and problem-example pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212–218.
- Wang, M., Yang, Z.-K., Liu, S.-Y., Cheng, H. N., and Liu, Z. (2015). Using feedback to improve learning: Differentiating between correct and erroneous examples. In *2015 International Symposium on Educational Technology (ISET)*, pages 99–103. IEEE.
- Wieringa, R. (2009). Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, pages 1–12.
- Yaacoub, E. E., Groves, R. M., Dawy, Z., Fowler Jr, F. J., Couper, M. P., Lepkowski, J. M., Singer, E., and Tourangeau, R. (2004). *Survey Methodology*, volume 337. John Wiley & Sons.
- Yamaguti, M. H., de Oliveira, F. M., Trindade, C. A., and Dutra, A. (2017). Ages: An interdisciplinary space based on projects for software engineering learning. In *Proceedings of the 31st Brazilian Symposium on Software Engineering*, pages 368–373. ACM.