A B A K Ó S
PUC Minas Instituto de Ciências Exatas e Informática

Design Patterns in Practice from the Point of View of Developers*

Padrões de Projeto na Prática do Ponto de Vista dos Desenvolvedores

Bruno Luan de Sousa[1]
Mívian Marques Ferreira[2]
Mariza Andrade da Silva Bigonha[3]
Kecia Aline Marques Ferreira[4]

Resumo

Padrão de projeto é um relevante tema de pesquisa que tem sido empiricamente investigado pela academia nos últimos anos. Entretanto, existe uma lacuna acerca da percepção do uso dos padrões de projeto na prática. Este artigo visa preencher essa lacuna por meio de uma análise de um cenário real brasileiro. Considerando que Belo Horizonte é uma das principais cidades brasileiras no contexto de desenvolvimento de software, conduziu-se um *survey* com 58 desenvolvedores de software ativos dessa cidade. Os resultados relatados neste artigo mostram uma real percepção do uso de padrão de projeto neste importante polo de desenvolvimento de software. Neste *survey*, identificou-se que padrões de projeto não estão amplamente disseminados nas indústrias locais, uma vez que 40% dos participantes afirmaram não utilizá-los frequentemente. Além disso, observou-se que a não aplicação de padrões de projeto está principalmente associado à falta de conhecimento pelos desenvolvedores e à falta de incentivo das empresas. Por fim, foram discutidos alguns benefícios apontados pelos participantes e listados os padrões de projeto mais usados e os menos usados pelos desenvolvedores.

Palavras-chave: Padrão de Projeto. *Survey*. Engenharia de Software.

*Submetido em 15/02/2019 - Aceito em 17/03/2020

[1]MSc in Computer Science at Computer Science at Federal University of Minas Gerais. Bachelor degree in Computer Science at the Federal Institute of Science, Technology and Education Southeast of Minas Gerais – bruno.luan.sousa@dcc.ufmg.br

[2]MSc in Computer Science at Computer Science at Federal University of Minas Gerais. Bachelor degree in Computer Science at Pontifical Catholic University of Minas Gerais – mivian.ferreira@dcc.ufmg.br

[3]Ph.D. in Computer Science at Pontifical Catholic University of Rio de Janeiro. MSc in Computer Science at Computer Science at Federal University of Minas Gerais. Associate Professor of the Computer Science Department at the Federal University of Minas Gerais – mariza@dcc.ufmg.br

[4]Ph.D. in Computer Science at Federal University of Minas Gerais. MSc in Computer Science at Computer Science at Federal University of Minas Gerais. Professor of the Department of Computing at Federal Center for Technological Education of Minas Gerais – kecia@decom.cefetmg.br

Abstract

Design patterns are a relevant research topic that has been empirically investigated by academia in the last years. However, there is still a gap in the perception of the use of design patterns in practice. In this paper, we aim to bridge this gap by analyzing a real Brazilian scenario. Considering that Belo Horizonte is one of the main Brazilian cities in the context of software development, we decide to carry out a survey with 58 active developers from this city. The results exhibited in this paper bring a real perception of the use of design patterns in a relevant center of software development. In this survey, we have identified that design patterns are not widely disseminated in the local industry since 40% of the participants claimed do not frequently make use of them. We have also found that the lack of use of design patterns is mainly associated with the lack of knowledge of these solutions by the developers besides the absence of incentives from the companies. Moreover, we discussed some benefits pointed out by the participants and listed the design patterns most used and less used by the developers.

Keywords: Design Pattern. Survey. Software Engineering.

1 INTRODUCTION

Design patterns are general solutions applied to recurring problems in the context of software design (GAMMA et al., 1994). They are recognized as good programming practices since they encourage the use of structures composed by inheritance, composition, and polymorphisms to make the communication between flexible objects and to reduce the coupling between modules. The most known and used design patterns are those described by the Gang of Four's (GoF) book (GAMMA et al., 1994).

Design patterns have been an important research topic in the Software Engineering industry and academia since 1994. Studies have pointed out the positive and negative aspects of them. For instance, some studies have showed the efficiency of these techniques for improving the internal quality of systems (BECK et al., 1996; PRECHELT et al., 2001; KERIEVSKY, 2004; CHRISTOPOULOU et al., 2012; NAHAR; SAKIB, 2015, 2016; ZAFEIRIS et al., 2017). Other studies (CARDOSO; FIGUEIREDO, 2015; JAAFAR et al., 2013, 2016; WALTER; ALKHAEIR, 2016; SOUSA et al., 2017, 2018, 2019) have identified co-occurrence between design patterns and complex structures called bad smells (FOWLER; BECK, 1999). Although design patterns are well-known solutions, and they have been empirically investigated, there is still a gap regarding the perception of the use of design patterns in real scenarios.

To bridge this gap, we surveyed 58 active developers from Brazil to identify the perception of the use of GoF design patterns in a real scenario. Brazil is an essential country in software development and has the 9^{th} major software market in the world, with a domestic market of US$ 18.6 billion (ABES, 2017). We chose Belo Horizonte as a real scenario because it is one of the main Brazilian cities in the context of software development, with hundreds of IT companies concentrated in it (AMARO, 2014; DANIELE, 2014). The results of our study indicate the design patterns are not so widely disseminated in the industry since 40% of the participants claimed do not frequently apply them. Analyzing these 40% participants' answers, we identified two types of developers: (i) the ones that do not often use design patterns because they do not know them or have low knowledge on these techniques, and (ii) those that work in companies that develop software for itself. Besides, they pointed out five main factors that discourage them from using these solutions: lack of knowledge, lack of the incentive of the companies, lack of documentation, overengineering and overthinking for adapting them to the problem context, and lack of pre-defined test. We also identified the design patterns benefits, and the ones most and least used by the developers.

The remainder of this paper is organized as follows. Section 2 provides a background about design patterns and presents the ones most common in the literature. Section 3 presents some related works. Section 4 presents the research questions and the method applied to carry out this study. Section 5 presents an overview of the survey. Section 6 shows the results of this study, answering the research questions. Section 7 shows the threats to validity and discusses the main decisions to mitigate them. Section 8 concludes this paper with the final remarks and indication of future work.

2 BACKGROUND

Design patterns were defined by Gamma et al. (1994) as descriptions of communications between classes and objects that are personalized to solve a general problem for a given context. These solutions allow developing flexible and extensible software with a high level of reuse. These structures are considered good programming practices since they improve software comprehension.

Design patterns play an essential role in the software development process. According to Cline (1996), one of the main advantages in the use of these solutions is the fact that they coordinate the development process and act as a standard vocabulary of communication among developers. Design patterns may be used as a type of documentation tool and help new developers of a team to understand the fragments of design from a software system. Besides, design patterns increase software reliability since they consist of proven architectures and accumulated experience.

On the other hand, the use of design patterns also provides some disadvantages for the software development context. Cline (1996) point out that one of the main disadvantages of the design patterns is the difficulty of learning them. Some design patterns are have a very complex structure. Besides, the misuse of design patterns in the software development may generate some problems, such as co-occurrences with bad smells, which make the software structure more complex and increase the costs with refactoring and maintenance (JAAFAR et al., 2013; CARDOSO; FIGUEIREDO, 2015; JAAFAR et al., 2016; WALTER; ALKHAEIR, 2016; SOUSA et al., 2017, 2019).

Although there are a lot of design patterns available in the literature, the most popular were proposed by Gamma et al. (1994). They built a catalog composed of 23 design patterns, which are known as Gang-of-Four (GoF). The design patterns introduced in this catalog are classified into three different categories: creation, structural, and behavioral. We present a description of each GoF category and design pattern as follows.

- **Creation:** it consists of abstracting the creation process of objects to create them in a suitable manner to each situation.
  - **Abstract Factory:** it provides an interface for creating a family of objects without specifying their concrete classes.
  - **Builder:** it separates creation and representation of a complex object, so that, different representations can be created during the object construction.
  - **Factory Method:** it defines a single interface for creating objects, but postpones the instantiation to the subclasses.
  - **Prototype:** it specifies the types of objects to be created via prototypes instances and creates new objects by copying the existing prototypes.
  - **Singleton:** it defines a single class instance and provides a global point of access.

- **Structural:** it consists of identifying simple ways to define the relationships between entities.
  - **Adapter:** it converts classes interfaces so that incompatible interfaces work together.
  - **Bridge:** it separates abstraction and implementation of the systems so that these two parts can vary independently.
  - **Composite:** it organizes objects in a hierarchy tree to represent part-whole. This decision allows the objects to be treated uniformly and individually.
  - **Decorator:** it allows to delegate additional features to objects dynamically.
  - **Facade**: it provides a single interface for a set of interfaces in a subsystem.
  - **Flyweight:** it uses data sharing to efficiently support large number of small objects.
  - **Proxy:** it provides an object that controls the access to the features of a given object.
- **Behavioral:** this type of design patterns consists of attributing responsibility to the entities and facilitating the communication between objects.
  - **Chain of Responsibility:** it defines a chain structure with objects to give an opportunity of more than one object treat a request.
  - **Command:** it encapsulates a request as an object, allowing to parameterize a client with different requests, log requests and support operations that can be undone.
  - **Interpreter:** this design pattern defines a representation and use it to interpret sentences of a given language of a grammar.
  - **Iterator:** it consists of a way to sequentially access elements of an object, hiding its structure.
  - **Mediator:** it defines an object to encapsulate the way how a set of objects interact one another.
  - **Memento:** it captures and externalizes an internal state of an object so that it can be restored to that state later.
  - **Observer:** it defines an one-to-many dependency between objects so that when an object changes, all its dependent objects are notified and updated.
  - **State:** it allows the object to change its behavior when its internal state changes.
  - **Strategy:** it defines a family of algorithms, encapsulates each one, and make them interchangeable.
  - **Template Method:** it defines a skeleton of an algorithm in a operation, postponing some steps for subclasses.
  - **Visitor:** it represents an operation to be run on the elements of an object structure.

3 RELATED WORK

This section provides an overview of some related works and discusses the main differences between them and the present work.

Design patterns emerged in 1994. Since then, many works have studied them in many aspects. For instance, Wendorff (2001) investigated some negative impacts of design patterns in software quality. They found that the lack of comprehension of design patterns and the wrong choices are the main factors that decrease their structural quality. In the same line, McNatt e Bieman (2001) and Izurieta e Bieman (2013) also identified some other factors that decay the quality of these solutions. According to them, strongly connected and highly dependents design patterns hinder the modularization of the system and increase the coupling in its internal structure.

Khomh e Gueheneuce (2008) carried out a study more specific than Wendorff (2001), McNatt e Bieman (2001) and Izurieta e Bieman (2013). They evaluated the impact of GoF design patterns on some internal attributes, such as expansion, capacity, simplicity, reuse, learning, among others. They identified that design patterns do not always improve the quality of these attributes and also pointed the Flyweight as the one that decreases the software quality.

Our study differs from the Wendorff (2001), McNatt e Bieman (2001), Khomh e Gueheneuce (2008), and Izurieta e Bieman (2013) because we have investigated the use of design patterns in practice and found factors that inhibit the use of design patterns by analyzing opinions of active developers and maintainers in the software industry.

The studies carried out by Wendorff (2001), McNatt e Bieman (2001), Khomh e Gueheneuce (2008) and Izurieta e Bieman (2013) encouraged other studies verifying the co-occurrence between design patterns and bad smells. Bad smells are design problems and require code refactoring (FOWLER; BECK, 1999). Many studies have investigated and identified co-occurrences between GoF design patterns and bad smells (JAAFAR et al., 2013; CARDOSO; FIGUEIREDO, 2015; JAAFAR et al., 2016; WALTER; ALKHAEIR, 2016; SOUSA et al., 2017, 2019). According to them, design patterns do not avoid the occurrence of bad smells, and the misuse of design patterns and the scattering and crosscutting concerns are the factors that have caused these co-occurrences in the software.

A systematic literature mapping carried out by Sousa et al. (2018) also investigated relations between design patterns and bad smells in the literature. They have identified that the misuse or inappropriate design pattern application, misuse planning from the system, and excessive assignment of functionality to the internal components from design patterns have contributed to introducing co-occurrences between design patterns and bad smells. The studies regarding co-occurrence between design patterns and bad smells have identified situations that have helped to reduce the quality of design patterns. The present work aimed to contribute to factors that inhibit the adoption of design patterns in the software development context in practice. We have also analyzed their use in practice.

Finally, other studies have analyzed the benefits and inhibitors of design patterns adop-

tion in software development activity. For instance, Cline (1996) studied the benefits and inhibitors of patterns application. He cataloged a total of seven benefits and three inhibitors and discussed each one of them. Hahsler (2008) evaluated the use of design patterns in the context of open source software developments. They analyzed the development process of almost 1,000 open source projects and concluded the adoption of design patterns in the open source context is influenced by the projects' profile and activity of the developers since projects with large team and developers who create the most of the code are more likely to adopt design patterns. Riehle (2011) analyzed the use of design patterns in large IT projects from three different companies and provides an interesting discussion of the contributions and benefits of these solutions. Riehle (2011) indicates that design patterns play a crucial role in the design, implementation, and documentation of the systems. Besides, design patterns benefit the evolution of the system with a shared vocabulary that allows passing system information from a developer to another. Zhang e Budgen (2013) evaluated which GoF design patterns are useful and not useful for software developments by analyzing the opinions of researchers on design patterns. They identified three useful design patterns: Abstract Factory, Composite, and Facade. Moreover, they pointed out that Flyweight, Interpreter, Memento, and Prototype are those less useful.

Santos et al. (2016) investigated the benefits of design patterns and the factors that contribute to their disuse. They identified the development process model, the lack of incentive of companies, and the short time to deliver the product are the main factors that influence the disuse of design patterns. The maintenance and code evolution, problem solution, and solution reuse are the main benefits provided by design patterns. Lano et al. (2018) investigated the use of model transformation (MT) design patterns in practice. They concluded although the use of MT design patterns is quite widespread, they have been used in an unconscious and unsystematic way because they are incorporated in some languages. They also warn that the documentation of MT design patterns and tool support need to be improved for this technique turn more popular.

The main contribution of this paper, in comparison with the related works, is the evaluation of the use of the GoF catalog in the real software development context. For instance, Cline (1996), Riehle (2011) , and Santos et al. (2016) focused only in benefits and inhibitors. Hahsler (2008), Zhang e Budgen (2013), and Lano et al. (2018) evaluated the use of design patterns and those most used and least used. However, the context and the design patterns evaluated by them are different from our study.

4 RESEARCH METHOD

According to Wohlin et al. (2012), a survey is "a kind of empirical strategy for collecting information about people for describing, comparing or explaining their knowledge and behavior". This paper presents a study of the use of design patterns in the Brazilian scenario through information from a set of active developers.

4.1 Research Questions

The research questions (RQn) to be analyzed are defined as follows.

RQ1: Have GoF design patterns been frequently applied in practice?

RQ2: When and why developers do not make use of design patterns?

RQ3: What are the main benefits of applying design patterns within a software project?

RQ4: Which GoF design patterns have been most used and least used by developers?

4.2 Design of the Survey

The questionnaire we constructed is based on a guideline defined by Kitchenham e Pfleeger (2008) and divided it into five question blocks, which are described as follows.

Consent Term. The first block consists of a consent term to explain the purpose of this survey for the participants and guarantee the anonymity of their answers. In this part, the participants need to agree with the rules to advance to the other blocks.

Personal Information. In this block, the participants need to enter with their personal information so that they can be identified. These participants' data are essential for avoiding duplicate answers and identifying developers that do not work in Belo Horizonte.

Background and Experience. This block aims to characterize the participants concerning their background and professional experience. For this purpose, we defined closed questions with options on the ordinal scale. These options refer to the characterization categories pre-defined by the authors.

Technical Block. This block collects technical information about the use of the design pattern by the participants in their company. This part is where the participants indicate the frequency that they apply design patterns, which design patterns they apply if these solutions help or not them during the software development process, and the benefits and factors that inhibit the use of these solutions in practice.

Final Block. The final block concludes the questionnaire with a comment for the participants suggesting some improvements or justifying their answers. We also dedicate a space for indications of email addresses from professionals that fit at the scope of this study. This block is optional for the participants.

4.3 The Survey Form

The methodology established to design the questionnaire is composed of five steps. Initially, we created an initial version following the structure described in Section 4.2. After, we

did an extensive review process where the questions and their options were intensely discussed among the authors to remove ambiguities and keep the statement of the questions as clear as possible.

Subsequently, we created a pilot questionnaire and asked for an active and expert developer to verify the clarity of the questions and the effectiveness of the options. We implemented the hints provided by him and built the final questionnaire, which is composed of 14 questions divided into five blocks, as described in Section 4.2. Table 1 provides an overview of the questionnaire.

The survey was implemented and managed via Google Forms[5]. We chose Google Forms because it allows us to collect and organize information through online forms. Besides, Google Forms is an open source, and it provides several resources to implement the questions, such as radio buttons, scales, among others. For each item created, we defined text with some hints and orientations to help the participants answer it and solve possible doubts.

4.4 Target Public

An essential element when we carried out a survey is to determine the population to be investigated. In this survey, we defined our target public as being professionals that work with software development and maintenance activities in companies from Belo Horizonte.

4.5 Participant Recruitment

We followed two approaches for recruiting participants. First, we created a list of known professionals and sent an email with the link to the questionnaire for them. In the survey, we made available a field where the respondents might indicate other developers. In this approach, we invited 40 professionals, including the indications.

To increase the number of respondents, in the second approach, we recruited people by posting the questionnaire on Facebook and disseminating it in a mailing list from UFMG and CEFET-MG. UFMG and CEFET-MG are universities from Belo Horizonte and have a lot of students who work in software development in companies.

5 SURVEY OVERVIEW

This section presents an overview of the process used in the survey and characterizes the companies as well as the participants of it.

[5]<https://www.google.com/forms/about/>

Table 1 – Overview of the questionnaire

	Consent Term	
1	Agreement with the consent term	
	Personal Information	
2	Request for participant's email	(Open question. This question is mandatory.)
3	Request for company where the participant currently works	(Open question. This question is mandatory.)
	Background and Experience	
4	Which is your degree?	(Technical; Undergraduate; Graduate)
5	About your professional experience in software development companies how much time of experience do you have?	(Up to 6 months of experience; Up to 1 year of experience; From 1 to 3 years of experience; More than 3 years of experience)
6	How do you evaluate your knowledge on design patterns?	(None; Low; Moderate; Expert)
7	What is the work focus of the company where you work?	(Only development of systems for other companies; Development and maintenance of systems for other companies; Development and maintenance of systems used by the company itself; Product development and maintenance; Other)
	Technical Block	
8	How often do you use design patterns to develop a system within your company?	(Never; Rarely; Usually; Always)
9	Which GoF design patterns have you applied for developing a system in your company?	(Each GoF design pattern was made available as an option. The participant might to check one or more option.)
10	Do you believe that design patterns provide any benefits in the software development process?	(Do not Help; Little Help; Fairly Help; Help a lot)
11	What are the main benefits provided by the design patterns in the software development process?	(Allows code reuse; Favor the comprehension of the system internal structure and the communication between developers; Allow greater system flexibility and ease of integration of new features; Favor future maintenance; Decrease the cost/effort in the development process; None; Other)
12	What are the main factors that inhibit the use of design patterns where you work?	(Lack of knowledge on these solutions; Pressure to deliver the product in a short time; Lack of planning from the development process; Undefined development process model; Complexity in learning these solutions; Low or missing availability of these solutions for some programming languages; Lack of domain in the used programming language; None; Other)
	Final Block	
13	Comment section	(Open question. This question is optional.)
14	Indication of other people	(Open question. This question is optional.)

Source: Elaborated by the authors.

5.1 General Overview

This survey was available from May 14^{th}, 2018 to June 11^{st}, 2018. In total, 62 developers answered the questionnaire being 20 of them contacted by the first recruitment approach, and 42 respondents by the second approach. Considering the first approach, we obtained 50% answer rate. For the second approach, we were not able to compute the answer rate since we do not have access to the number of subscribed in the mailing list, and it was not possible to calculate the number of people that our post reached on Facebook.

As we used social networks to recruit participants, we had to carry out a manual inspection of the participants' personal information to remove answers that could negatively impair the results of this survey. We sought for: duplicated answers, inactive developers or maintainers, and professionals from other cities. We described the steps of this inspection as follows.

Duplicated Answers. This step consists of analyzing the collected data and removing duplicated answers. We verified the email address in each one of 62 responses and identified three participants who answered the questionnaire twice. We discussed these cases and decided to keep the most recent answer from them. Therefore, after this step, we obtained 59 responses.

Inactive Professionals. This step aims to remove answers from developers that are not working with software development or maintenance. We analyzed the affiliations' names of each one of the 59 respondents. One of them claimed he is not working. So, we removed his data from this study and remained 58 answers.

Professionals from Other Cities. This step aims to remove the answers of professionals outside of Belo Horizonte. Here, we checked the name of the participants' affiliation again. We did not identify cases to be removed. This step remains with 58 responses.

In summary, steps 1, 2, and 3 received as input, in this order, 62, 59, and 58 answers from participants. At the end of these steps, we removed four records resulting in a total of 58 responses. We analyzed and summarized them to answer the research questions.

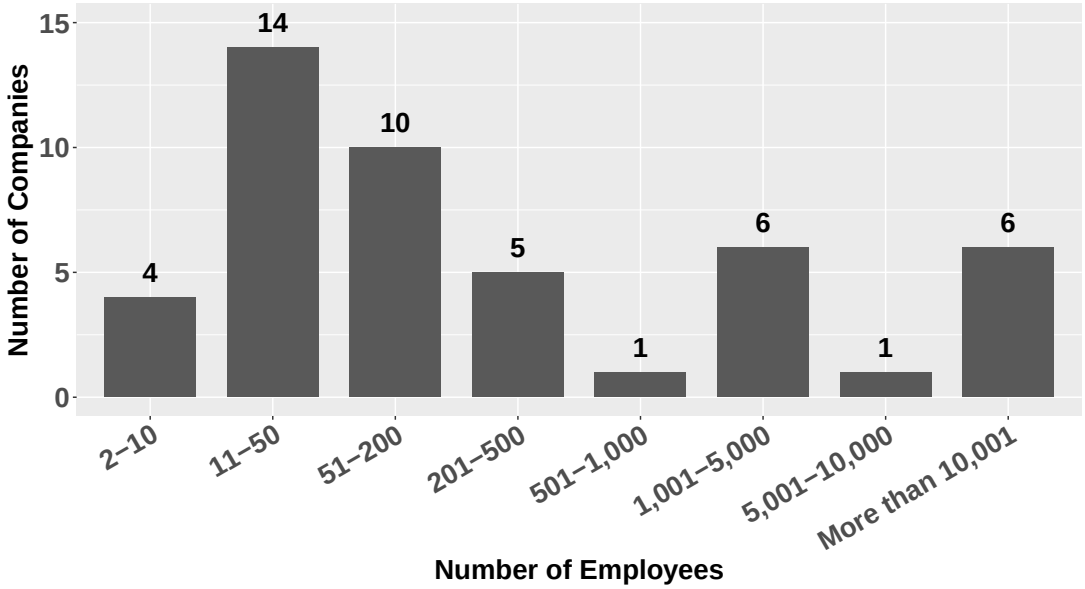
5.2 Companies Overview

One of the questions in the personal information block asked about the current participants' affiliation. To provide an overview of the respondents, we classified the companies according to their available size on LinkedIn[6]. We chose LinkedIn because it is a business social network with thousands of companies registered and a large variety of information about them. Figure 1 shows the distribution of the participants according to their company size.

Each classification had at least one affiliation regarding it. The most common companies have from 11 to 50 employees. Some companies were reported more than once since there are participants who work in the same companies. So, we summarized all affiliations only once in Figure 1, and therefore, 58 participants represented 47 companies. Although we have obtained

[6]<www.linkedin.com>

Figure 1 – Distribution of the companies size



Source: Elaborated by the authors.

information regarding the developers of several companies, it is important to highlight that the focus of this paper is to analyze the view of developers regarding the use of design patterns in practice.

5.3 Participants' Background and Experience

The third block aimed to collect information about the participants' background and experience to characterize them. The first question asked about the education degree of each respondent. To answer this question, the respondent should choose one of the following options: (i) Technical, (ii) Undergraduate, and (iii) Graduate.

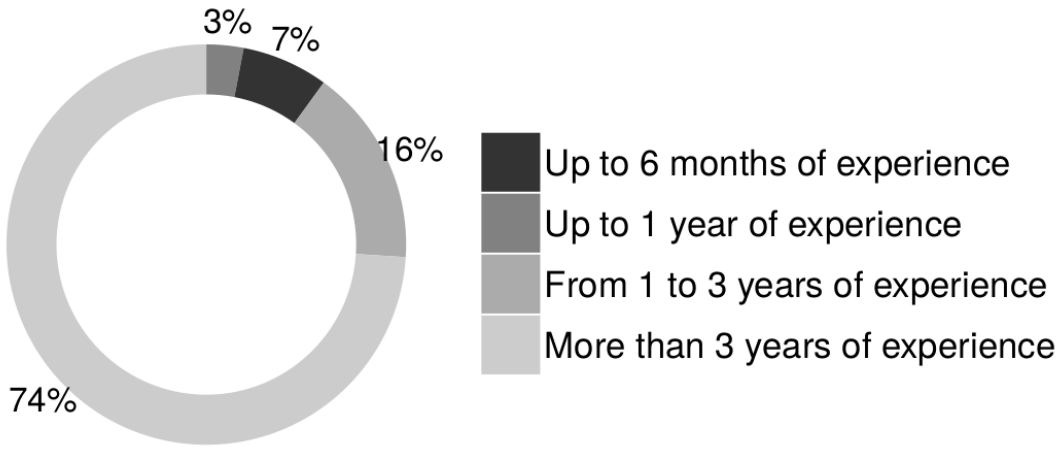
The technical degree is a type of professional education existing in the Brazilian educational level that is offered to students who are enrolled or have finished high school. This degree aims to provide technical qualification at the high school according to the professional profile chosen, and therefore, allow quick integration of the student into the job market (SENAI, 2013).

This study was carried out with graduate and undergraduate professionals. They correspond to 52% and 46% of our population, respectively. We also obtained an answer from one technical professional who represents 2%.

To evaluate the participants' professional experience, we asked them to inform the time of experience with software development or maintenance through the options previously provided. Most of the participants in this survey, 74%, are expert professional with more than three years working in software companies. We also had some participants, 7%, that claimed to be at the beginning of their career. The remainder of our population answered to have "up to one year of experience", and "from one to three years of experience" representing 3% and 16%

respectively. Figure 2 shows the distribution of the participants' professional experience.

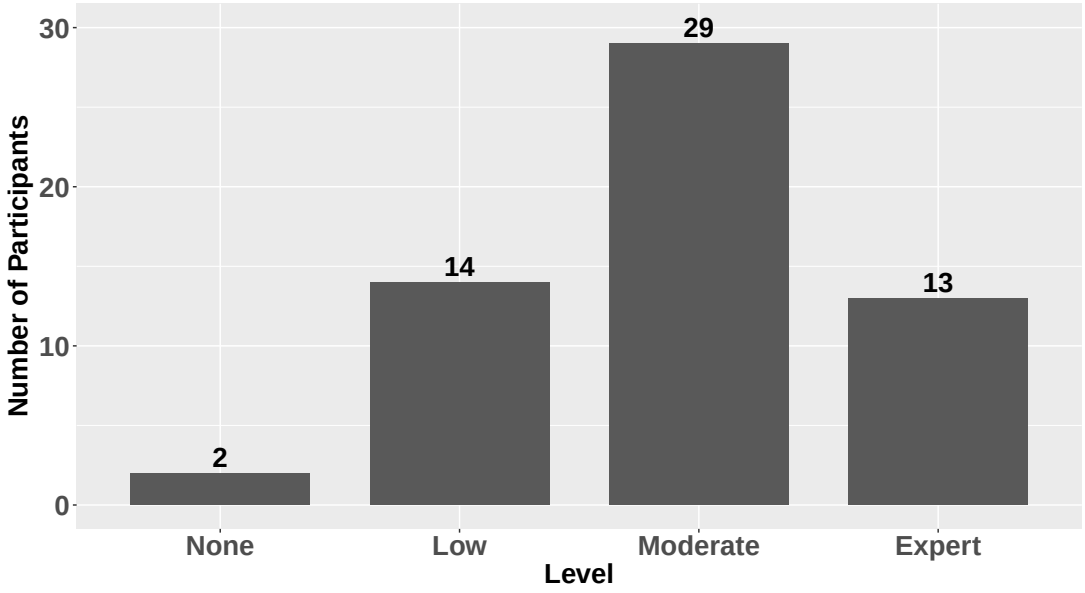
Figure 2 – Participants' level of professional experience



Source: Elaborated by the authors.

Finally, we characterized the participants regarding their knowledge of design patterns, asking them how they classify their knowledge of these techniques. Most of the population, 72%, reported to have a moderate knowledge or to be an expert on design patterns. However, a significant part of the participants, 28%, reported to have low or none k nowledge. Figure 3 summarizes these information.

Figure 3 – Participants' knowledge level on design patterns



Source: Elaborated by the authors.

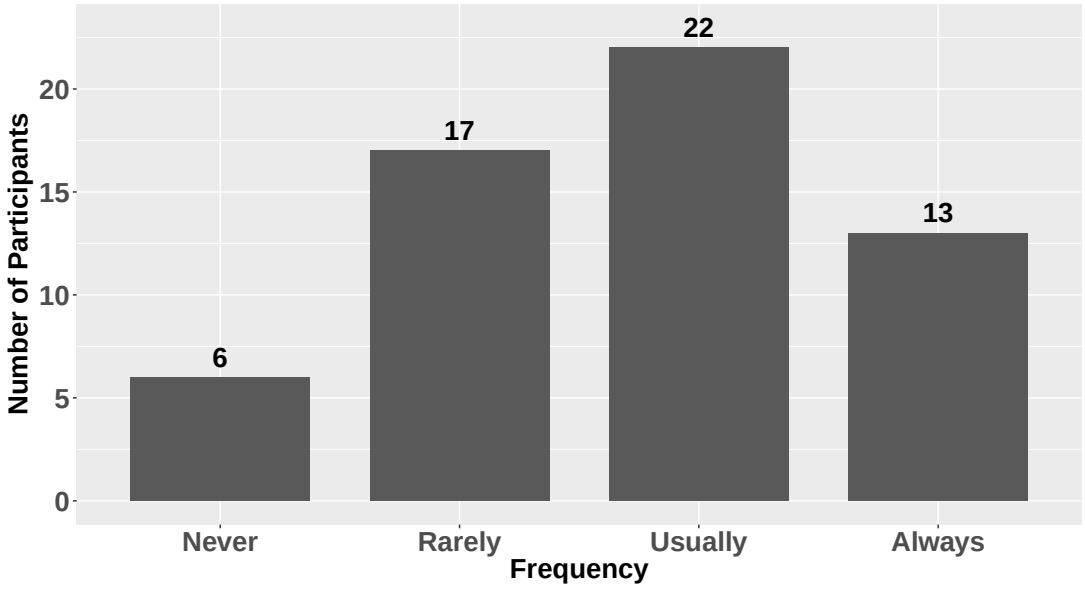
6 RESULTS

This section presents the results and answers the research questions.

RQ1: Have GoF design patterns been frequently applied in practice?

With the RQ1, we aim to analyze the current use of the GoF design patterns in software companies and verify if they have been frequently applied in software development. So, to answer this research question, we asked the participants the following question: "How often do you use the GoF design patterns to develop software in your company?". We summarized the answers in Figure 4.

Figure 4 – Frequency of the use of design pattern by participants



Source: Elaborated by the authors.

Summary of RQ1. We identified that most of the participants, 60%, know and have frequently applied GoF design patterns during the software development. However, a high rate of our population, 40%, claimed that rarely or never had used these techniques. This insight indicates that although design patterns have been pointed out as good programming practices and their use have been encouraged, they are not widely disseminated in the software industry.

RQ2: When and why developers do not make use of design patterns?

With the RQ2, we aim to identify the profile of the participants that do not apply design patterns and find out the reasons that inhibit this disuse. To answer this question, we analyzed only answers from the participants that rarely or never used design patterns during software development or maintenance activities. Therefore, in this research question, we examined a total of 23 participants' answers.

Out of the 23 participants' answers analyzed in this research question, 52% of the participants do not know them or have low knowledge of them. This result shows a lack of knowledge on design patterns is one of the main factors that inhibit their use. It also reinforces the fact that design patterns are not so well disseminated in the software industry, as some studies have pointed out (SPEICHER, 2013; WAGEY et al., 2015). Moreover, we may conclude that it is mandatory to improve the teaching of design patterns in undergraduate courses to disseminate them in the software industry.

Besides, we identify that the profile of the software development activity in a company might influence the use of design patterns. For instance, companies that develop software for their use tend not to apply design patterns. In contrast, companies that build and maintain software for other companies tend to use design patterns.

We also analyze if the development process adopted by the companies has influenced the use of design patterns in the practice. The participants' answers reported seven different types of the development process. They are: Waterfall, Extreme Programming (XP), Iterative and Incremental, Kanban, Scrum, Mix of Scrum/Waterfall, and Scrum with DevOps. Besides, some participants have indicated that their company does not have a well-defined model, and others claimed that the process changes among projects. We confronted the participants' answers about the development process with the information of frequency of use of design patterns. We did not identify any pattern that indicates an influence of the development process model used by the company on the use of design patterns.

Regarding the reasons that make them inhibit the use of design patterns, they reported: pressure to deliver the product in a short time, lack of planning from development activity, and undefined development process model are the main factors responsible for this inhibition. These factors reflect an absence of incentives from companies concerning the use of design patterns. One participant highlighted that the implementation of design patterns demands time to create the solutions and increase costs for the companies that are rarely willing to pay for these costs.

Another factor pointed out by the participants is the lack of documentation of the systems, mainly because the specification is an essential step in the software development process. According to the participants, most companies do not document their systems. This lack of documentation turns the comprehension of the internal software structure difficult and may induce the developer not to use design patterns and introduce complex structures in the systems.

Some participants highlighted that design patterns require an overengineering and time studying how to adapt the solutions so that they fit into the problem context. This situation turns the developer less productive. Besides, even with the high dedication of the developer, there is a risk of implementing the design pattern wrongly. The lack of predefined tests to verify and validate them in the development process also discourages developers to apply design patterns since, in their conception, design patterns are complex and hard to implement, and may impair the code comprehension.

Summary of RQ2. We conclude that most of the developers that do not apply design patterns are those who do not know or have low knowledge of these solutions. The profile of

software development activity also influences their use. Regarding to the reasons that impact this disuse, the main factors reported by the participants were: (i) lack of the incentive of the companies to use these solutions, (ii) lack of documentation for the systems, (iii) overengineering and overthinking for adapting them to the problem context, and (iv) lack of pre-defined test to verify and validate their implementations.

RQ3: What are the main benefits of applying design patterns within a software project?

With the RQ3, we aim to identify the benefits provided by the use of design patterns in software development. To answer this question, we considered only responses from the participants that claimed to have moderate knowledge or be an expert professional on design patterns.

Table 2 summarizes the benefits. By the participants' answers, we may conclude that: (i) flexibility in future maintenance, (ii) the improvement at the legibility and communication between developers, and (iii) the code reuse are practically common sense between the professionals in the software industry since more than 75% of the participants indicated these three factors as benefits of the use of design patterns. Some developers also pointed out that the flexibility and ease of including new features in software during its life cycle is another important benefit. Another important benefit indicated by the participants was the cost/effort reduction in the development process. According to the participants, design patterns demand costs, and programming efforts during the development of the software. However, these costs and programming efforts tend to decay during the life cycle of the systems. Considering that most software life cycle consists of maintenance and enhancement (WAGEY et al., 2015), the use of design patterns may be a good strategy for companies to reduce their costs with software development.

Table 2 – Benefits provided by design patterns

Benefit	# Answers
Facilitate future maintanance	33
Favor the code comprehension and internal structure of the system and the communication between developers	33
Allow code reuse	32
Allow greater system flexibility and ease of integration of new features	27
Decrease the cost/effort in the development process	16
Other	3

Source: Elaborated by the authors.

The participants reported other benefits brought by design patterns. One of them is the practicality in solving recurring problems since they act as a starting point to model a solution

to the approached problem. They also work as implicit documentation in software because the developer builds the internal structure of the software based on the solution documented in the literature. Finally, one participant reported that deficiencies in some programming languages make it difficult to create solutions for some problems. So, design patterns may act as an additional artifice and help developers to create exciting solutions for these problems.

Summary of RQ3. Table 2 answers this research question. It shows the flexibility in future maintenance, the improvement at the legibility and communication between developers, and code reuse are practically common sense among the developers. Also, the use of the design pattern favors the inclusion of the new feature in the software and decrease the cost and effort during the maintenance phase. Finally, the participants pointed out some additional benefits. They reported design patterns to act as a starting point to solve common problems, act as implicit documentation in software, and may act as an additional artifice and help developers avoid deficiencies from some programming languages.

RQ4: Which GoF design patterns have been most used and least used by developers?

With the RQ4, we aim to identify the GoF design patterns most used and least used in practice. To answer this research question, we asked the participants: "Which GoF design patterns have you applied for developing software in your company?", and made available the GoF design patterns as options. They could choose more than one design pattern. Table 3 summarizes the participants' answers.

The Singleton was the most cited by the participants, 67%. This result shows it is a significant design pattern and has helped developers to create efficient solutions while solving problems. Moreover, we identified that the three of the five design patterns most used, Singleton, Factory Method, and Abstract Factory, are classified as creation. This result shows the most concern of the developers to develop software is abstract the process of creating objects to make it flexible and facilitate the inclusion of new features during the software life cycle.

Summary of RQ4. Table 3 answers this research question. The five design patterns most used are Singleton, Factory Method, Observer, Abstract Factory, and Facade. In contrast, the five design patterns least used are Flyweight, Visitor, Memento, Interpreter, and Bridge. We also identified a high rejection regarding the Flyweight design pattern, since any participant did not mention it.

7 THREATS TO VALIDITY

This section discusses some threats to validity based on Wohlin et al. (2012).

Internal Validity. We collected data via a questionnaire. One of the drawbacks of surveys is that inconsistencies are only revealed after the data collection period, and this fact may represent a threat to validity. To mitigate it, after collecting the data, we manually inspected the answers to remove inconsistencies and responses that did not fit on the scope of our study. Each found inconsistency was discussed among the authors before deleting it.

Table 3 – Design patterns most used by participants.

Design Pattern	# Answers	% Use
Singleton	39	67%
Factory Method	26	45%
Observer	23	40%
Abstract Factory	22	38%
Facade	21	36%
Iterator	19	33%
Builder	18	31%
Adapter	17	29%
Decorator	16	28%
Chain of Responsibility	13	22%
Proxy	13	22%
Template Method	13	22%
Composite	12	21%
Strategy	12	21%
Prototype	7	12%
Command	6	10%
Mediator	6	10%
State	5	9%
Bridge	4	7%
Interpreter	3	5%
Memento	3	5%
Visitor	2	3%
Flyweight	0	0%

Source: Elaborated by the authors.

To characterize the companies, we classify each one concerning their number of employees. This classification may impact the results because of the classification scheme and process. To mitigate this threat, we used a classification scheme of a significant business social network called LinkedIn, which also provides information about the classification of the companies' size. Since LinkedIn are continually updating their information on companies' size, we decided to use these data to characterize the companies identified in this survey.

We use social networks to find developers and increase the number of participants in our survey. However, using social networks may bias our results, since we miss the control of the population. To mitigate this threat, we proposed two approaches for recruiting participants. Initially, we obtained a controlled sample with developers who we sent the survey via email. After that, we used Facebook and mailing lists to disseminate our survey. We based on the participants' answers to validate their profile and remove the ones that did not fit into our study.

One confounding variable is the comprehension of the questions when the participants are answering the questionnaire. To ensure well comprehension, we carefully reviewed all questions to remove the ambiguities. We also created a pilot questionnaire and asked an expert professional to validate the clarity of the questions and the effectiveness of the options. Finally, we introduced a brief text in each question with some hints and descriptions about its focus.

External Validity. Although we had a large number of participants with different backgrounds, we may not generalize our results since this survey is limited to developers from Belo Horizonte. However, the found results are important and reveal the reality regarding the use of design patterns in companies from a relevant center of software development. Such findings support to propose solutions for improving the use of design patterns in similar scenarios.

Construct Validity. The creation of the questionnaire needs to be well defined to remove ambiguities. So, we have used an extensive methodology for reviewing the questionnaire items. The subjects were initially discussed by the authors to validate their clarity, and after that, we have requested an active developer for validating it too. We have considered the feedback provided by the professional and based on them to improve our questionnaire and mitigate this threat.

Conclusion Validity. Most questions we have created in the questionnaire were closed questions. This decision may be considered a threat to validity because it may influence the participants' answers. To mitigate it, we have performed an exhaustive discussion to create options that best reflect the industry reality. Moreover, we have created an open option in some questions to allow the participants to expose other opinions and factors that were not made available as options.

8 CONCLUSION

This paper aimed to identify the perception of Brazilian developers on the use of the GoF design patterns in practice. We collect information from the participants via a questionnaire regarding the research questions. We analyzed answers from 58 active developers and maintainers from Belo Horizonte.

Initially, we have identified that the design patterns are not widely disseminated in the companies of this relevant Brazilian center of software development since 40% of the participants in this survey claimed not frequently apply design patterns in the software development process inside their affiliations. The main reason pointed out to justify this disuse was the lack of knowledge of design patterns. This fact indicates that the teaching of design patterns need to be improved in undergraduate courses. They also pointed out the lack of incentive of the companies, the lack of documentation, overengineering and overthinking for adapting design patterns to the problem context, and lack of the pre-defined test to verify and validate the correctness of the implementation, as other factors that discourage them from using these solutions. Although 40% of the participants do not frequently use design patterns, all participants believe these solutions bring some benefits. Benefits as code reuse, improvement at the legibility and communication between developers, and flexibility in future maintenance are practically common sense among the participants. Advantages as favors the inclusion of the new feature, decrease cost and effort during the maintenance phase, practicality in solving common problems, documentation implicit, and additional artifice that help to supplement deficiencies from

programming language are other benefits indicated by the participants that may encourage the use of these techniques in practice. Finally, we identified the design patterns most used and least used for developers. The five design patterns most used are Singleton, Factory Method, Observer, Abstract Factory, and Facade. And the five design patterns least used are Flyweight, Visitor, Memento, Interpreter, and Bridge.

Therefore, we can summarize the main contributions of this paper as follows:

- The use of design patterns is not widespread in a relevant Brazilian center of software engineering.
- Most of the developers that do not apply design patterns are those who do not know or have low knowledge of these solutions. Besides, situations as lack of the incentive of the companies to use these solutions, lack of documentation for the systems, overengineering and overthinking for adapting them to the problem context, and lack of a pre-defined test to verify and validate their implementations are also factors that contribute for the disuse of the design patterns in practice.
- The main benefits provided by the design patterns in practice are flexibility in future maintenance, improvement at the legibility and communication between developers, code reuse, ease in the inclusion of the new feature in the software, and cost and effort reduction in maintenance activities. Besides, design patterns act as a starting point to solve common problems, work as implicit documentation in software, and serve as an additional resource helping developers to avoid deficiencies from some programming languages.
- Singleton, Factory Method, Observer, Abstract Factory, and Facade are the design patterns most used by Brazilian developers in practice. In contrast, Flyweight, Visitor, Memento, Interpreter, and Bridge are the design patterns least used.

As future works, it is important to (i) extend this research for other scenarios and Brazilian cities to depict the use of design pattern in the Brazilian scenario as a whole; (ii) investigate if software process certification is a factor that encourages developers and companies to apply design patterns during the software development; and (iii) build a tool for helping developers to implement design patterns automatically in practice.

References

ABES. **Brazilian Software Market**: Scenario and trends. São Paulo, 2017. 28 p. (In portuguese).

AMARO, M. **Do you know the São Pedro Vallery in Belo Horizonte?** April Publisher. 2014. Available at: https://exame.abril.com.br/carreira/conhece-o-bairro-de-sao-pedro/. Accessed on July, 2018. (In portuguese).

BECK, K. et al. Industrial experience with design patterns. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 18., 1996, Berlin. **Proceedings [...]**. Berlin: IEEE, 1996. p. 103–114.

CARDOSO, B.; FIGUEIREDO, E. Co-occurrence of design patterns and bad smells in software systems: An exploratory study. In: BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, 11., 2015, Goiania. **Proceedings [...]**. Goiania: SBC, 2015. p. 347–354.

CHRISTOPOULOU, A. et al. Automated refactoring to the strategy design pattern. **Information and Software Technology**, v. 54, n. 11, p. 1202–1214, nov. 2012.

CLINE, M. The pros and cons of adopting and applying design patterns in the real world. **Communications of the ACM**, New York, v. 39, n. 10, p. 47–49, oct. 1996.

DANIELE, A. **Top 10 Brazilian cities that hire IT professionals**. April Publisher. 2014. Available at: https://exame.abril.com.br/carreira/as-10-cidades-do-pais-que-mais-contratam-profissionais-de-ti/. Accessed on July, 2018. (In portuguese).

FOWLER, M.; BECK, K. **Refactoring: Improving the Design of Existing Code**. 1. ed. Boston: Addison-Wesley Professional, 1999.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-oriented Software**. 1. ed. Boston: Addison-Wesley Professional, 1994.

HAHSLER, M. A quantitative study of the adoption of design patterns by open source software developers. In: TAN, F. (Ed.). **Global Information Technologies**: Concepts, methodologies, tools, and applications. 1. ed. Pennsylvania: Igi Global, 2008. p. 560–577.

IZURIETA, C.; BIEMAN, J. A multiple case study of design pattern decay, grime, and rot in evolving software systems. **Software Quality Journal**, v. 21, n. 2, p. 289–323, feb. 2013.

JAAFAR, F. et al. Analysing anti-patterns static relationships with design patterns. **Electronic Communications of the EASST**, Berlin, v. 59, n. 1, p. 1–26, 2013.

JAAFAR, F. et al. Evaluating the impact of design pattern and anti-pattern dependencies on changes and faults. **Empirical Software Engineering**, v. 21, p. 896–931, mar. 2016.

KERIEVSKY, J. **Refactoring to Patterns**. Boston: Pearson Higher Education, 2004.

KHOMH, F.; GUEHENEUCE, Y. Do design patterns impact software quality positively? In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 12., 2008, Athens. **Proceedings [...]**. Athens: IEEE, 2008. p. 347–354.

KITCHENHAM, B.; PFLEEGER, S. Personal opinion surveys. In: SHULL, F.; SINGER, J.; SJØBERG, D. (Ed.). **Guide to advanced empirical software engineering**. 1. ed. London: Springer, 2008. p. 63–92.

LANO, K. et al. A survey of model transformation design patterns in practice. **Journal of Systems and Software**, v. 140, p. 48–73, jun. 2018.

MCNATT, W.; BIEMAN, J. Coupling of design patterns: Common practices and their benefits. In: INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE ON INVIGORATING SOFTWARE DEVELOPMENT, 25., 2001, Chicago. **Proceedings [...]**. Chicago: IEEE, 2001. p. 574–579.

NAHAR, N.; SAKIB, K. Automatic recommendation of software design patterns using anti-patterns in the design phase: A case study on abstract factory. In: CEUR WORKSHOP, 3., 2015, New Delhi. **Proceedings [...]**. New Delhi: CEUR, 2015. p. 9–16.

NAHAR, N.; SAKIB, K. Acdpr: A recommendation system for the creational design patterns using anti-patterns. In: INTERNATIONAL CONFERENCE ON SOFTWARE ANALYSIS, EVOLUTION, AND REENGINEERING, 23., 2016, Suita. **Proceedings [...]**. Suita: IEEE, 2016. p. 4–7.

PRECHELT, L. et al. A controlled experiment in maintenance comparing design patterns to simpler solutions. **IEEE Trans. Softw. Eng.**, Piscataway, v. 27, n. 12, p. 1134–1144, dec. 2001.

RIEHLE, D. Lessons learned from using design patterns in industry projects. In: NOBLE, J. et al. (Ed.). **Transactions on pattern languages of programming II**: Special issue on applying patterns. 1. ed. Berlin: Springer, 2011. p. 1–15.

SANTOS, M.; SOUZA, M.; FIGUEIREDO, E. Design patterns in java: A practical study on their use and benefits. In: WORKSHOP ON SOCIAL, HUMAN AND ECONOMIC ASPECTS OF SOFTWARE, 1., 2016, Maceió. **Proceedings [...]**. Maceió: SBC, 2016. p. 31–40. (In portuguese).

SENAI. **Technical of High School**. 2013. Available at: https://www.senaiac.org.br/index.php/educacao/tecnico-de-nivel-medio.html. Accessed on January, 2019 (In portuguese).

SOUSA, B.; BIGONHA, M.; FERREIRA, K. Evaluating co-occurrence of gof design patterns with god class and long method bad smells. In: BRAZILIAN SYMPOSIUM ON INFORMATION SYSTEMS, 13., 2017, Lavras. **Proceedings [...]**. Lavras: SBC, 2017. p. 396–403.

SOUSA, B.; BIGONHA, M.; FERREIRA, K. A systematic literature mapping on the relationship between design patterns and bad smells. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 33., 2018, Pau. **Proceedings [...]**. Pau: ACM, 2018. p. 1528–1535.

SOUSA, B.; BIGONHA, M.; FERREIRA, K. An exploratory study on cooccurrence of design patterns and bad smells using software metrics. **Software: Practice and Experience**, v. 48, n. 7, p. 1079–1113, may 2019.

SPEICHER, D. Code quality cultivation. **Communications in Computer and Information Science**, Berlin, v. 348, p. 334–349, 2013.

WAGEY, B.; HENDRADJAYA, B.; MARDIYANTO, M. A proposal of software maintainability model using code smell measurement. In: INTERNATIONAL CONFERENCE ON DATA AND SOFTWARE ENGINEERING, 2., 2015, Yogyakarta. **Proceedings [...]**. Yogyakarta: IEEE, 2015. p. 25–30.

WALTER, B.; ALKHAEIR, T. The relationship between design patterns and code smells: An exploratory study. **Information and Software Technology**, v. 74, p. 127–142, jan. 2016.

WENDORFF, P. Assessment of design patterns during software reengineering: Lessons learned from a large commercial project. In: EUROPEAN CONFERENCE ON SOFTWARE MAINTENANCE AND REENGINEERING, 5., 2001, Lisbon. **Proceedings [...]**. Lisbon: IEEE, 2001. p. 77–84.

WOHLIN, C. et al. **Experimentation in Software Engineering**. 1. ed. Berlin: Springer, 2012.

ZAFEIRIS, V. et al. Automated refactoring of super-class method invocations to the template method design pattern. **Information and Software Technology**, v. 82, p. 19–35, feb 2017.

ZHANG, C.; BUDGEN, D. A survey of experienced user perceptions about software design patterns. **Information and Software Technology**, v. 55, n. 5, p. 822–835, may 2013.