




On the Use of UML in the Brazilian Industry: A Survey

Ed Wilson Júnior  [Universidade do Vale do Rio dos Sinos | edwjr7@edu.unisinos.br]

Kleinner Farias  [Universidade do Vale do Rio dos Sinos | kleinnerfarias@unisinos.br]

Bruno da Silva  [California Polytechnic State University | bcdasilv@calpoly.edu]

Abstract Over the past decade, UML modeling has been used in the industry in software development tasks, such as documenting design decisions and promoting better communication between teams, as pointed out in recent studies. However, little is known about the factors, practitioners' perceptions, and practices that affect UML use in real-world projects. This article, therefore, reports exploratory research focused on investigating how UML is used in practice in the Brazilian software industry. In total, 376 professionals from 210 information technology companies answered an online questionnaire about the factors affecting use, difficulty and frequency of use, perceived benefits, and contextual factors that prevent the adoption of UML models. In addition, 20 professionals participated in a semi-structured interview answering basic questions about professional experience, vision on software modeling, use of tools, and other aspects of UML. The main results show that: 74% of the participants answered that they do not use UML frequently. Factors such as (1) high time pressure to develop features; (2) the cost of disseminating a common model understanding among diverse audiences and; (3) the difficulty of evaluating the quality of the models affect the effective use of UML. In general, most participants know UML, but do not use it frequently (or do not use at all) in their projects. Finally, this article draws some challenges, implications and research directions that can be explored in upcoming studies for promoting UML modeling in practice.

Keywords: UML, Unified Model Language, Practice, Industry, Survey

1 Introduction

UML models can play a crucial role in software development tasks such as documenting design decisions and promoting better communication within and across teams (OMG, 2017). Some previous studies (Bucchiarone et al., 2021; Fernández-Sáez et al., 2018; Chaudron et al., 2012) highlight that the use of UML modeling can provide benefits to the software development process, such as providing a common understanding among team members, understanding the details of design decisions, and ultimately making the process more efficient after all. However, in practice, such benefits are often overlooked or not observed. Some studies (Fernández-Sáez et al., 2018; Chaudron et al., 2012; Störrle, 2017) argue that such benefits can be realized when there is a consistent and (in)formal application of modeling, where developers typically use UML throughout the project and have precise control over its use. As we can rarely find such a scenario, researchers (Fernández-Sáez et al., 2018; Petre, 2014) have tried to draw a clear picture of UML use in real-world projects.

Today, the current literature (Akdur et al., 2021; Fernández-Sáez et al., 2018; Petre, 2014; Chaudron et al., 2012) lacks a broad and exploratory understanding of practitioners' perceptions of the factors that affect or even compromise the adoption of UML modeling in real-world projects. More specifically, little is known about how practitioners deal with software modeling in the Brazilian software development industry context. Previous studies (Petre, 2014, 2013) have focused on collecting opinions from participants to understand which UML diagrams are most used. However, this assumes that participants' perceptions and experiences worldwide match those at the regional level (i.e., country or significant geographic region). These studies neither explore, for example, whether the project

context can influence the UML adoption nor do they discuss practitioners' views on the perceived usefulness of UML itself.

This article investigates the state of the practice regarding the use of UML in the Brazilian industry by surveying and interviewing software practitioners in that country. Specifically, this work seeks to investigate (1) how practitioners use UML and (2) the relevance of its use in real-world software projects. Therefore, this study surveyed 376 professionals from 210 Brazilian information technology companies. We selected participants based on two criteria: (1) level of knowledge and practical experience related to software modeling; and (2) programming experience in regular projects. Participants answered an online questionnaire about their experience with UML, the difficulties of adopting it, factors that affect its practical use, frequency of use, and the benefits it brings (or could bring). Also, in the second phase, we interviewed 20 participants following a semi-structured interview protocol to understand further the survey results.

Our findings are encouraging and bridge the literature gap regarding the impact of the organizational culture issue in UML use, the analysis of factors that hinder UML use, and help to understand the broader landscape of the UML adoption. Some evidence already reported in the literature is reinforced. This study can help companies and software practitioners understand the broader landscape of UML use, thus supporting their future decision-making around software practices and techniques in their future projects. Academia and industry can benefit from our insights on how to improve their software modeling practices or develop new tools and processes. Besides, this study also benefits researchers and practitioners by providing additional empirical knowledge about practical issues concerning UML modeling in a broader view.

This article is an extended version of our previous work (Júnior et al., 2021) in several ways. First, the article underwent a careful review and was significantly improved as a whole. Second, the research protocol was improved by adding the list of interview questions and considering the location of the companies where the participants work. Third, the number of survey participants increased from 314 to 376 (i.e., 62 new participants), new findings are generated from this sample, and more thorough discussions regarding six research questions. In addition, this article presents additional discussions, identifies open challenges and implications, and describes the key underlying issues that need to be addressed in future investigations.

The article is structured in seven sections. Section 2 defines related work. Section 3 details the adopted methodology. Section 4 describes the results for each research question. Section 5 brings up qualitative reflections and insights for future work. Section 6 presents the main threats to the study's validity. Section 7 wraps up the article and include some ideas for future work.

2 Related Work

The selection of related works was performed based on two steps: (1) initial search in digital repositories, such as Google Scholar¹ and Scopus², was done to identify articles regarding the UML usage and survey in this research field; and (2) filtering of the selected articles considering the alignment of such works with the objective of our article (Section 3.1). We selected studies from 2014 until now as our study is based on the findings reported in (Petre, 2014). After that, they were analyzed (Section 2.1) and compared to nine studies to identify research opportunities (Section 2.2).

2.1 Analysis of Related Works

Petre (2014). This work performed an empirical study about the use of UML in practice, which involved interviews conducted over two years with more than fifty software developers. The participants were mainly from North America and Europe, but some were from Brazil, India, and Japan, and many had worked in more than one country. Petre found that participants did not use UML universally but used it consistently in specific contexts such as embedded systems (e.g., automotive, aerospace, etc.). In addition, Petre reported that the UML models are not used homogeneously, on the contrary, the interviewees reported heterogeneity in relation to the way to use the models in practice. Typically, interviewees assumed different roles throughout the development cycle, using UML models differently in each role. Petre also reported that the way practitioners used UML diagrams depended on the problem domain faced.

Ozkaya and Erata (2020). This research involved 109 professionals from 34 countries, representing the different profiles, positions, types of software projects, and years of experience to understand how professionals use UML to

model software architecture from different viewpoints: functional, information, concurrency, development, deployment, and operational. They found that the information and functional viewpoints are the most popular ones. Moreover, the obtained results showed that most participants (88%) used UML when they needed to model system architecture from different viewpoints.

Fernández-Sáez et al. (2015). This study presents a survey on the use of UML in software maintenance. They surveyed 178 practitioners working on software maintenance projects in 12 different countries. Their results indicate that companies can improve system maintenance by leveraging the use of UML diagrams while executing maintenance tasks; however, it would require a significant effort to update UML diagrams as source code evolves.

Farias et al. (2018). We reported research findings on a shorter survey to identify the UML use in practice in the Brazilian industry. Two hundred and twenty-two practitioners from 140 different Information Technology companies answered a questionnaire concerning their experiences with UML, the difficulty in adopting it, and what should be done to increase adoption in practice. The results show that: (1) only 60 participants (28.2%) had used UML in their daily work; (2) 55.41% of the surveyed participants did not disagree with the statement that UML is the “lingua franca” in software modeling; (3) 61.26% reported to find that the automatic creation of UML diagrams to represent a big picture of the system under development would be useful to boost UML use.

Ciccozzi et al. (2019). This work carried out a systematic review that involved 63 research studies and 19 tools from more than 5400 initial entries. The objective was to identify, classify, and evaluate the existing solutions for UML model execution (i.e., automatically interpret or translate models into running software). The main results of this study are: (1) there is a growing scientific interest in the execution of UML models; (2) model-level debugging is supported in very few cases; (3) only a few surveys provide evidence of industrial use, with very limited empirical assessments; and the most common limitation is the coverage of the UML language.

Störrle (2017). This article conducted an online survey involving 82 professionals to determine whether and to what extent they use conceptual models and for what purposes. Specifically, the author sought to grasp (1) if practitioners use UML and BPMN (Business Process Modeling Notation) for software modeling; (2) for what purposes are these modeling languages used; (3) what are the different ways of using these models in practice; and (4) how often practitioners use these modeling languages. Störrle found that models are perceived to be widely used by study participants, and UML is the leading language. Störrle reported three distinct usage modes of models, of which the most frequent is informal usage for communication and cognition.

Fernández-Sáez et al. (2018). This study performed a case study in a multinational company's ICT department and involved 31 interviews with employees who work on software maintenance projects. The study mainly focused on the use of UML in software maintenance. They found that using software modeling notations such as UML is considered beneficial for software maintenance but needs to be tailored to

¹<https://scholar.google.com/>

²<https://www.scopus.com/>

its context. The authors also provided a list of recommended practices that contribute to the increased effectiveness of software modeling.

Ho-Quang et al. (2017). The authors conducted a large-scale survey with 485 responses from contributors from 458 different open source projects. In that context, they found that collaboration was the most important motivation for using UML in open source projects as teams use UML during communication and planning of joint implementation efforts. UML models seem to benefit new contributors' onboarding but do not seem to be a significant factor in attracting new contributors.

Neto et al. (2021). This study presents an overview of the adoption of UML in IT companies in São Carlos (Brazil) and the region through a survey of 21 questions answered by 24 participants. Also, it aims to compare how language is taught in universities. The results show a significant use of UML, including in companies that adopt agile methods and the authors suggest that the content on UML is preserved in the curriculum of educational institutions, in an updated and optimized way, meeting the trends presented by IT companies. The study also points out that the opportunities in the area of modeling, with the mastery of agile methodologies and the trend of continuous acceleration of processes, are vast. One of them would be, at first, the adequacy of UML modeling for agile methodologies, without the most valued asset in these methodologies: time.

2.2 Comparative Analysis and Opportunities

Six Comparison Criteria (CC) were defined to assist in identifying similarities and differences between the proposed work and the selected articles. This comparison is crucial to identify research opportunities using objective rather than subjective criteria. We describe the six comparison criteria below:

- **Context (CC01):** Studies that involved professionals in the Brazilian industry.
- **Participant profile (CC02):** Studies that collected participant data for screening and profile characterization.
- **Specific geographic region (CC03):** Works that explored the UML use in a specific regional scope.
- **Applicability of UML (CC04):** Studies that evaluated which factors prevent the adoption of UML in the software industry.
- **Interviews with participants (CC05):** Studies that triangulated quantitative and qualitative data.
- **Different domains (CC06):** Studies that involved software developers working in different problem domains or business segments.

Table 1 compares the selected papers and summarizes whether they meet the criteria completely, partially, or do not meet, thus contrasting them with our work. Moreover, it highlights the similarities and differences between them. We observe that only our work fulfill's all criteria. In this sense, two research opportunities were identified: (1) few studies broadly inspect the adoption of UML models from the perspective of the Brazilian industry; and (2) no study produced empirical evidence from a survey and conducting inter-

views at the same time. The next section outlines a methodology to explore these identified research opportunities.

Table 1. Comparative analysis of the selected related works

Related Work	Comparison Criterion					
	CC1	CC2	CC3	CC4	CC5	CC6
Proposed Work	●	●	●	●	●	●
Petre (2014)	◐	○	○	●	○	●
Ozkaya and Erata (2020)	◐	●	○	○	○	●
Fernández-Sáez et al. (2015)	○	●	○	○	◐	●
Farias et al. (2018)	●	○	●	◐	○	●
Ciccozzi et al. (2019)	○	○	○	●	○	◐
Störrle (2017)	○	●	○	●	○	◐
Fernández-Sáez et al. (2018)	○	●	○	○	◐	○
Ho-Quang et al. (2017)	○	●	○	●	○	●
Neto et al. (2021)	●	●	●	●	○	○

● Completely Meets ◐ Partially Meets ○ Does not attend

3 Methodology

This section presents the research methodology followed for conducting our survey. This protocol was formulated based on well-known guidelines (Wohlin et al., 2012; Kitchenham and Pfleeger, 2008) to design and run empirical studies, as well as based on our experience in carrying out previous surveys (Farias et al., 2018; Júnior et al., 2021). This section is organized as follows. Section 3.1 introduces the main objective and research questions. Section 3.2 describes the adopted experimental process. Section 3.3 describes the questionnaire and interview formulated and applied in the study.

3.1 Objective and Research Questions

The study objectives are twofold: (1) to understand the diffusion and relevance of the use of UML in the Brazilian industry; and (2) to analyze at what level developers understand the benefits of UML in real-world projects. We formulated six research questions (RQ) to analyze different facets of these objectives. Table 2 describes the formulated RQs.

3.2 Experimental Process

Figure 1 introduces the adopted experimental process, composed of three phases discussed as follows:

Phase 1: Selection of participants. Participants were selected based on the following criteria: level of knowledge, practical experience related to software modeling, and programming in industrial software development projects. Using these criteria, we sought to select participants with academic backgrounds and practical experience in the Brazilian industry. This set of all possible participants represents the target population (Kitchenham and Pfleeger, 2008; Wohlin et al., 2012). More specifically, the target population comprises practitioners working in Brazil — including developers (different seniority levels), software architects, and project managers — with academic backgrounds obtained from Brazilian universities. This population represents those who are in a position to answer the questions asked and to whom the research results apply (Kitchenham and Pfleeger,

Table 2. Research questions investigated in this article

Research Questions	Motivation	Variable
RQ1: What factors influence the effective use of the UML?	Reveal the influencing factors in a broader usage of UML models in practice.	Usage-influencing factors
RQ2: What makes UML modeling a challenging practice?	Understand the challenges practitioners face that hinder the adoption of UML modeling	Adoption-hindering
RQ3: What benefits do practitioners realize when it comes to using UML?	Reveal the most commonly realized benefits when using UML modeling.	Perceived benefits
RQ4: How often do practitioners use UML?	Understand how often practitioners use UML modeling.	Frequency of use
RQ5: How does the context of software projects limit the use of UML in organizations?	Identify context factors that limit the use of UML in organizations.	Project context
RQ6: How do practitioners view UML modeling?	Reveal the practitioners' vision regarding the adoption of UML modeling.	Practitioner view

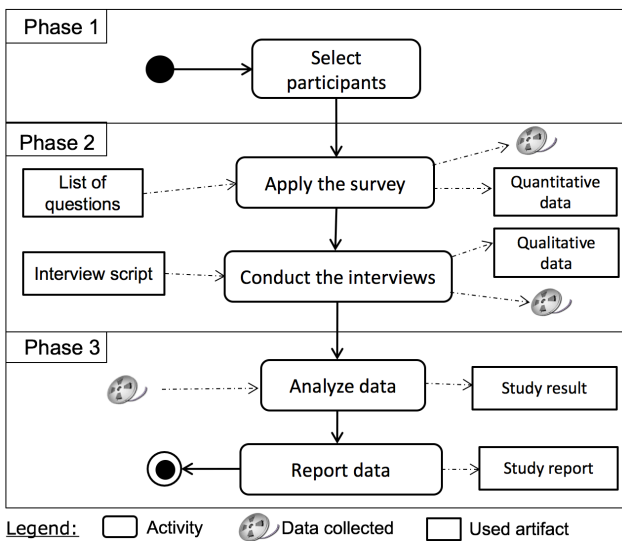


Figure 1. Experimental process

2008; Wohlin et al., 2012). In total, 376 participants answered the questionnaire.

Phase 2: Application of the questionnaire and interviews. This phase focused on the application of the questionnaire and the interviews execution. We conducted interviews to collect additional qualitative data related to research questions. Such data is essential to triangulate the obtained results (Section 4) from our questionnaire and interviews. The questionnaire (discussed in Section 3.3) was sent by e-mail to the target population, totaling more than 406 people invited. In total, the study had 376 participants. We carefully selected the target population to avoid collecting data from people with inadequate profiles. We invited undergraduates, graduate students (master's and doctorate), industry professionals with a recognized academic background, and professionals identified in the social network of professionals, such as LinkedIn. The 376 participants worked in 210 companies in different Brazilian regions (midwest, south, southeast, and northeast). After completing the stages of answering and sending the questionnaire, we randomly invited 27 participants (out of 376) for a semi-structured interview (Wohlin et al., 2012; Farias et al., 2015). 20 participants, namely (P1-20) hereafter, accepted the invitation. The script was direct, starting from basic questions about the professional experience, the vision of software modeling, the use of tools, and other aspects of UML. The interviews were performed and recorded using the Microsoft Teams software. In a further step, we triangulated the qualitative and quantitative data

from the interview and the questionnaire to explore complementary aspects of the data.

Phase 3: Data analysis. This phase sought to analyze the data collected through the questionnaire and interviews carefully. For this, we first analyzed the collected data (interviews and survey) separately and then compared them (triangulation). Initially, we analyzed the data collected through the survey and tabulated it. Then, we used those initial survey results as the basis to formulate the interview questions. Therefore, the interviewees answered questions that sought to explore the results obtained through the survey more deeply, seeking consistency in the data analysis. The investigation provided interaction through a dialectical process, interaction, and reflection between the researcher and the participants. We manually performed interview data analysis and went from a broad view to a more focal one without divergences. That helped us obtain complementary evidence to explain the quantitative results and then derive concrete conclusions from a chain of evidence formed from the systematic alignment of quantitative and qualitative data.

3.3 Questionnaire and Interviews

Data were collected from interviews and an online questionnaire³ (created in Google Forms). The study repository⁴ has more information. Participants reflected on their experience on UML software modeling in practice through our semi-structured interviews. Table 3 presents the list of questions used in the interview. These interviews helped us to enrich the body of qualitative data. The authors ask a list of predefined questions for all respondents. New questions were formulated based on the answers given by the participants. We chose the online survey instrument because it enabled quick application, and fast distribution, thus reaching a larger number of individuals in geographically diverse locations at no additional cost. The survey questions examined research gaps in previous studies and apprehended the structures of the previously developed questionnaire. In addition, we based the design of the questionnaire and interview questions on the findings reported by Petre (2014).

³ Questionnaire: <https://forms.gle/TFRwsgJ7UFUcpcfN7>

⁴ Study repository: <https://github.com/edwjr/surveyQuestionnaire>

Table 3. List of questions used in the interview

ID	Question
q1	Which company do you currently work for?
q2	What is your view on software modeling?
q3	How is UML used where you work?
q4	What is the main difficulty in using UML?
q5	Why do developers tend not to use UML in organizations?
q6	When is the use of UML worth it?
q7	Do you use any specific software modeling tools to visualize and edit diagrams?
q8	How often do you not consult the software documentation and work directly with source code?
q9	How much effort do you put into reading UML diagrams?
q10	What improvements should be made to enhance the use of UML?

4 Results

This section presents the obtained results concerning the formulated research questions (described in Section 3.1). We used histograms to provide an overview of the collected data from the responses of 376 survey participants and 20 interviews.

4.1 Analysis of the participants' profile

Table 4 summarizes the participant's profile, reporting different facets including education, undergraduate degree, job role, overall experience, professional experience with software modeling, experience with software development, and location. The 376 participants who responded to the survey came from 210 companies in Brazil (at the time of data collection). As some questions were not required, the sum (n) is not necessarily equivalent to the total number of participants (376).

Education. The majority (68.1%) either had already graduated from college (36.9%) or were pursuing a degree as a student, while 10.6% had already completed either a post-graduate specialization (7.9%) or a Master's degree (3.7%) in the field of computing. 20.6% of the participants were "certified technicians" in the field of computing⁵. Only one participant did not earn an undergraduate degree in computing but rather mathematics, subsequently pursuing a master's degree in applied computing. Regardless of their level of education, all participants were professionals with experience in the industry.

Undergraduate degree. Most participants (91.8%) had an undergraduate degree in computing. In Brazil, universities offer computing degrees under different names, including systems analysis (51.9%), computer science (28.7%), and information systems (11.2%). This shows our participant pool has a strong academic background which complements the participants' practical experience. Considering their job roles, 50.7% were software developers, 23.6% were systems analysts and 2.4% were software architects. Software architects

⁵In Brazil, some schools have programs to offer high school degrees with an additional professional/technical certificate.

Table 4. The profile data of the participants.

Characteristic (n=376)	Answer	#	%
Education	Technical Certificate	77	20.6%
	Undergraduate student	117	31.2%
	Graduate	138	36.9%
	Specialization	22	7.9%
	Master	14	3.7%
Undergraduate degree	System Analysis	195	51.9%
	Computer Science	108	28.7%
	Information Systems	42	11.2%
	Others	31	8.2%
Position	Developer	187	50.7%
	Systems Analyst	87	23.6%
	Software Architect	9	2.4%
	Manager	7	1.9%
	Others	79	19.6%
Overall Experience	< 2 years	138	37.5%
	2-4 years	129	35.1%
	5-6 years	56	15.2%
	7-8 years	10	2.7%
	> 8 years	18	4.9%
Professional experience with software modeling	< 2 years	227	61.2%
	2-4 years	91	24.5%
	5-6 years	25	6.7%
	7-8 years	10	2.7%
	> 8 years	18	4.9%
Professional experience with software development	< 2 years	126	34.1%
	2-4 years	120	32.5%
	5-6 years	54	14.6%
	7-8 years	28	7.6%
	> 8 years	41	11.1%
Geographical distribution of companies	Northeast	3	1%
	Midwest	31	15%
	South	102	42%
	Southeast	13	6%
	More than one location	61	29%

and managers accounted for 1.9% of the sample. Thus, 80% of the participants were in job positions directly related to software development practices.

Overall experience. The experience level is diverse in our participant pool, showing higher concentration in the 2 to 6 years range (62.5%), 7.6% had seven years or more of overall professional experience.

Modeling experience. Regarding the characteristics of modeling experience, participants were experienced, but not highly, with software modeling. The expected result would be the lack of experience since previous empirical studies point to low adoption of UML models in the industry. About 38% of the participants had more than two years of professional experience in software modeling, while the others said they had less than two years of experience.

Development experience. Regarding software development, overall, participants reported more years of experience compared to software modeling experience (when software modeling is considered a separate activity). As expected, practitioners are generally more exposed to experience programming tasks than modeling tasks. That is why we see more years of experience in "software development" than "software modeling" when these are considered "separate activities".

Geographical distribution of companies. Regarding work location, our participants came from 210 different companies located in all regions of the country except the northern region. The largest concentration was in the southern region with 102 companies, representing 42% of the sample. The midwest and southeast regions were 15% (31) and 6% (13), respectively, and the northeast region represented 1% (3). Companies located in more than one region represent 29% (61).

Given the participant demographics, we consider the

participants' profile adequate to answer the research questions of our study for two main reasons. First, the participants came from a diverse set of companies (210), avoiding responses biased by experiences obtained in a limited set of companies. Also, the large number of companies the participants came from increases the chances of participants with experiences in diverse business contexts and organizational cultures, thus improving the quality of the signal we can get in the study. Second, all the participants had some formal education in computing, thus increasing the chances that they had some level of training in software modeling. This reduces the risk of biasing their answers because they had not known UML or had not heard about software modeling before the survey. Moreover, the 20 interviewed participants reported modeling experience greater than five years, and they worked in software development in areas such as education (4 participants), agribusiness (3), e-commerce (2), government (3), trading (3), product exports (2), and finance (3). That diversity of areas, experience, and knowledge enriched the discussion. For ethical and privacy reasons, we chose not to present the names of the companies where participants worked. The following sections discuss the results obtained organized by research question.

4.2 RQ1: What factors influence the effective use of the UML?

Figure 2 presents the collected data concerning the UML usage-influencing factors (RQ1). We explored three factors to answer RQ1: (a) time pressure that leads developers not to do software modeling, focusing only on working on the code; (b) the cost of promoting a common model understanding among the involved people with different levels of education/experience; and (c) the difficulty in assessing the quality of the created models.

Time. Figure 2(a) indicates that 52% of the survey participants and 18 of the 20 interviewees reinforced that the short development time and high demands are the main factors that influence the use of UML since the software systems developed are getting larger and more complex every day due to the increasing demand of customers. *“Currently the projects are large and with a very short delivery time, you can barely deliver 100% software, imagine a documentation that would have to be updated at every step”* (P17). This also leads to complex software projects that cannot be easily managed by project stakeholders and cause software systems to be delivered late (or with budget overrun) or incorrectly developed (Ozkaya and Erata, 2020). Consequently, they end up opting for other complementary methods, such as screen prototyping, or not even creating UML models.

Cost of promoting understanding. Figure 2(b) shows that most of the participants either fully agree (34%) or partially agree (34%) that the cost of promoting a common understanding among team members is a significant influencing factor on UML use. Conversely, when we approached the interviewees with this question, most of them (12 out of 20) considered that the cost of promoting accurate modeling understanding between different people with different levels of education/experience and viewpoints is low, diverging from the survey data. This divergence possibly emerged

since most interviewees worked in teams where all members had the same level of experience/training, thus leading to a smoother alignment regarding model understanding. The academic skill set affects where/how stakeholders have learned software modeling, influencing their modeling approaches and their relevant practices through the modeling experience Akdur et al. (2017).

Difficulty evaluating. Figure 2(c) shows the difficulty in evaluating the quality of UML models is another significant usage-influencing factor (21% fully agree, 40% partially agree). Also, data from the interviews supported the difficulty in evaluating the models created and identified that this is one of the factors that affect the effective use of UML in the industry.

Moreover, the results on the usage-influencing factors support previous findings (Chaudron et al., 2012; Fernández-Sáez et al., 2015; Bucchiarone et al., 2021; Störrle, 2017). Bucchiarone et al. (2021) advocate that stakeholders model informally to support communicative and cognitive processes using emergent and flexible graphical notations in the early stages of the software development process. Störrle (2017) also indicates that informal modeling (e.g., sketching on a whiteboard) is considered more effective in promoting communication, collaboration, and understanding. However, it is worth noting that such diagrams can be scrapped or become inaccurate since they are not maintained together with the updated source code. Jackson (2019) points out that informal representations can be a good start for modeling, but it is limited, gives inconsistent interpretations, and cannot be analyzed mechanically. Additionally, previous experimental studies such as (Ho-Quang et al., 2017; Petre, 2014; Scanniello et al., 2014) revealed that some issues challenge UML's effectiveness. For instance, the UML complex notation as a whole, preference for other modeling approaches (e.g., informal sketches), and certain problem domains or industries might be more suitable than others for UML modeling. However, professionals have developed *ad hoc* practices that employ UML models in reasoning and communication about design, both individually and in collaborative dialogue.

On the other hand, in some scenarios and industries, models can be transformed into programs using the proper tools. In such cases, models have a longer service life and must be kept up to date. It is also often observed that different teams and sub-organizations within the same company can use different modeling approaches for different purposes at different stages of the software development lifecycle (Heldal et al., 2016). Therefore, either informal modeling or “traditional UML modeling” with automated code generation can become alternatives when time is a first-class constraint.

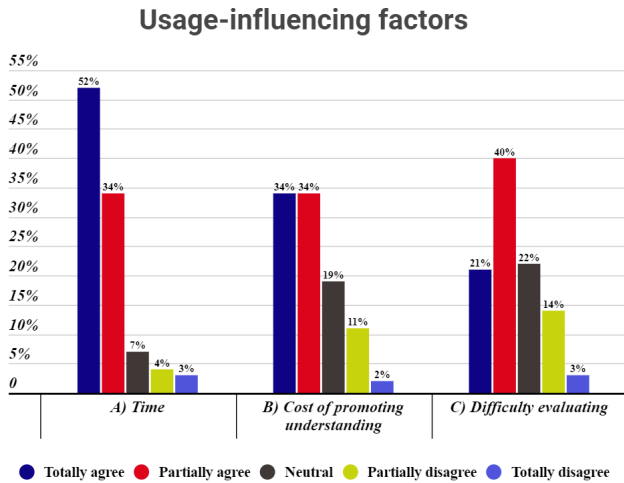


Figure 2. Usage-influencing factors (RQ1)

Summary of RQ1: *The results show that most participants indicate three points that affect the use of UML diagrams: (1) limited available time to create and maintain diagrams; (2) the cost of promoting proper understanding among different people with different levels of education/experience and viewpoints is high; and (3) difficulty in evaluating the quality of the diagrams. We understand that companies may need different modeling practices for different projects or roles within projects. Practitioners should consider those three points when considering UML modeling as part of their development processes.*

4.3 RQ2: What makes UML modeling a challenging practice?

Figure 3 shows the collected data regarding RQ2. From the survey responses, we highlight three adoption-hindering challenges: (a) the company's culture, which affects the way UML is used, (b) the necessary effort to keep different UML diagrams in sync, and (c) the high effort to create and maintain the models.

Company culture. Figure 3(a) indicates that 56% answered that they totally agree, 30% partially agree, and 10% were neutral. From the interviews, participants pointed out that, in some organizations, there is a culture of risking and failing as a path to learn quickly and meet customer needs, even if it requires much rework, thus, sometimes neglecting planning and upfront design. In addition, one of our interviewees mentioned: "I believe that the greatest difficulty is to change paradigms, especially when working with more mature teams that have grown without this modeling" (P4). Although the current state of practice has reached some degree of automation in systems engineering, its tasks still require many human resources. Thus, introducing process change in an organization already in operation is not easy (Böhm et al., 2014).

It is important to note that organizations may need different modeling approaches for different projects or even for different engineering roles within projects (Akdur et al., 2021). As also described in (Heldal et al., 2016), different

units within the same company tend to use different modeling approaches. In addition, in the same project, different engineers may use different modeling practices, depending on their tasks and responsibilities (Akdur et al., 2021).

Synchronization of diagrams. Figure 3(b) shows that 37% of the participants partially agree and 30% fully agree that keeping diagrams in sync is a significant challenge that hinders UML use, corroborating the majority of the interviewees (19). Although collaborative tools for software modeling exist, our result reinforces the findings reported in other studies conducted with industry participants (Chaudron et al., 2012; Cicchetti et al., 2016; Kuhn et al., 2012; Liebel et al., 2018), which appointed problems related to insufficient support for collaboration. There is a gap between UML tools and advanced solutions specialized in supporting collaboration. In addition, the next generation of modeling tools should support round-trip engineering to synchronize related UML diagrams and source code. Since modeling a software system's structural and behavioral aspects within a single model is not a trivial task, UML has proposed a set of diagrams to support a multiview modeling approach. Thus, different aspects of the system under development are represented by different diagrams.

High effort. Figure 3(c) revealed that 41% totally agree, 38% partially agree, 13% are neutral, 7% partially disagree, and 1% totally disagree. Therefore, the vast majority consider the effort invested in the creation and maintenance of UML models unanimously pointed out by the interviewees. "The biggest problem is the cost of keeping the diagrams as the system changes. In addition, it is still difficult to maintain a strong culture of maintenance and updating of models" (P17). Another interviewee complements: "from a maintenance point of view, I think that some improvements would be necessary for the diagrams to provide a better figure of the big picture, allowing to identify more quickly relevant issues such as impact and points that can be taken into attention" (P4). In Ozkaya and Erata (2020), the authors mentioned that modeling software architectures based on UML from the concurrency point of view has relatively less interest on the part of professionals. One important reason here could be UML's lack of support for modeling concurrency and race conditions. In addition, based on the findings of this study, most professionals are not used to planning development issues (e.g., source code organization and software construction and release processes) during the modeling and design, and this is usually omitted until the implementation. Interviewee 11 reports: "UML is used at the beginning of the project, more specifically the projection phase, but with the progress being left aside, it ends up being outdated, since most developers focus only on the code and management does not make large charges on its use" (P11). In this context, Fernández-Sáez et al. (2015) pointed out that the modeling tool used to maintain/modify UML diagrams is an important factor when deciding whether to use a software development process. There are different types of tools with different benefits: licensed tools (which implies an investment but also return with possible training, customizations, etc.) vs. open tools or specific tools for modeling in UML (which check the syntax correction) or general modeling tools (are more "accessible").

UML was identified as the dominant notation in Forward

and Lethbridge (2008). The authors found that UML modeling tools are primarily used for initial design, while UML is not widely used for code generation. The study participants seemed open to incorporating modeling into their processes. However, the difficulty of keeping models up to date with code changes is a significant depreciation factor (68% agreement on this from Forward and Lethbridge (2008)). The analysis performed on Forward and Lethbridge (2008) is particularly interesting, finding that programmers are more likely to agree that modeling tools are “heavy-weight.” Given this scenario, Fernández-Sáez et al. (2018) points out that it would be desirable to have a tool that would create and maintain documentation containing a mix of text and diagrams, in addition to having features that improve traceability between model and text to avoid leaving the documentation and the model out of sync. It would also be useful to have a tool that supports diagram versioning that matches the system version, searching model elements and presenting different views for the diagrams (for different consumers of information diagrams). In addition, another point we noted is that most participants are not used to putting effort into upfront planning and design (such as modeling) when they attempt to tackle coding issues.

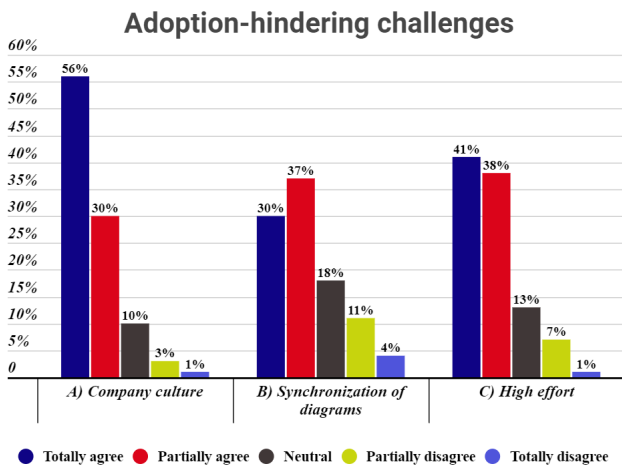


Figure 3. Adoption-hindering factors (RQ2)

Summary of RQ2: The results show that (a) organizational culture represents a significant challenge to enabling the adoption of UML models since the adopted engineering practices and the culture of agility sometimes do not give room to modeling. Therefore, we observe that modeling in agile processes consists of a unique pattern of UML use. (b) Synchronization between UML artifacts makes it difficult to use in highly collaborative software teams, and (c) the overall high effort to develop and maintain models is scarce in current organizational cultures.

4.4 RQ3: What benefits are realized when using UML?

Figure 4 shows a summary of collected data related to RQ3. We asked three questions related to (a) whether using UML selectively (only a few diagrams) helps to minimize complexity, avoid problems of completeness and inconsistency between diagrams, (b) whether UML models are helpful during application integration discussions, and (c) whether UML helps to form a common system understanding among developers.

Figure 4(a) indicates that 39% fully agree and 39% partially agree, and 15% are neutral. Figure 4 (b) shows that 49% fully agree; 41% partially agree, and; 7% are neutral. Figure 4 (c) reveals that 41% fully agree, 41% partially agree, and 11% are neutral.

All twenty interviewees unanimously agreed that using UML benefits software development, as it helps in the general understanding of the system context, thus facilitating communication in the team. “The use of this language enables the understanding and discussion of the architecture of a project by the entire team and allows the representing more complex and difficult flows” (P17). “UML is a powerful language for understanding software at various levels of abstraction. When used properly it contributes to creating a better product. When used improperly (in a forced way) ends up consuming resources and not helping much. In short, diagrams should be used as a means to understand various aspects of the software to be developed and not as the end. The goal of development is software and not diagrams” (P9). These factors are identified in Ho-Quang et al. (2017) where most participants (79%) found UML useful for understanding systems, improving communication between developers, guiding implementation, and managing project quality. Interviewees also mentioned UML could help with defect detection and design/implement integration of heterogeneous applications. However, inconsistent model interpretations can have serious consequences, especially when multiple and conflicting stakeholders are involved. For example, different interpretations between the development team, customers, and regulatory bodies can lead to rework, delays, and financial and legal repercussions. This risk may be exacerbated because compliance verification is usually performed later in the software development process. Consequently, any problem discovered in the compliance check (when applicable) is expensive to repair (Usman et al., 2020).

Participants of Petre (2014) reported using UML more enthusiastically, working in a more scope-focused manner, and keeping the artifacts manageable in size and suitable to avoid synchronization and consistency issues. The interest revolves around problem-solving or decision-making to avoid undue costs. One area that deserves further research is how the use of UML is shaped by the context of the domain - an investigation that requires much more access to a variety of software industries.

This context demonstrates that it is necessary to understand what actually facilitates effective software development. All this evidence highlights the need to consider the relationship of tools, including notation, both with the community of practice and with the application domain. Partici-

pants reinforced the fact that software developers are open to understanding the concepts and that, at the same time, they want to use tools that make the process effective. Otherwise, they tend to discard them if they are at odds with their practices.

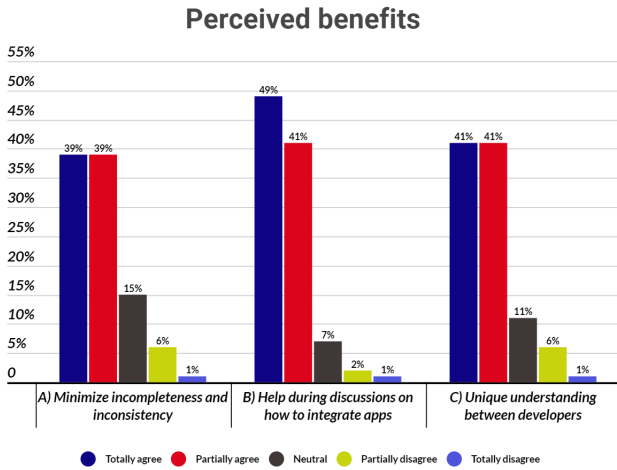


Figure 4. Perceived benefits (RQ3)

Summary of RQ3: *Selectively using only a few UML diagrams helps minimize complexity and avoid problems of completeness and inconsistencies between diagrams. In participants' view, using UML is beneficial and can help avoid issues in the project, enabling better system understanding and assisting in integration discussions.*

4.5 RQ4: How often is UML used?

Figure 5 presents the participants' responses on the use of UML in their work. As the question was not mandatory, 365 of the 376 participants answered it. 74% answered that they do not use UML frequently, while 26% answered that they use UML quite often. This result reinforces findings in Ozkaya and Erata (2020), in which the authors report that 35 of the 50 subjects in the study do not use UML in practice. Similarly, Gorschek et al. (2014) found that practitioners do not frequently use UML. When they do it, they do it informally, with minimal or no tool support, and the notation is not necessarily enforced to be UML.

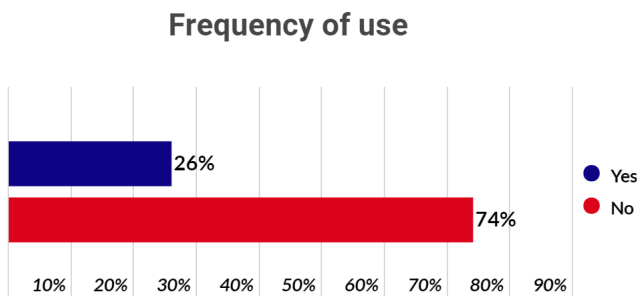


Figure 5. Frequency of use (RQ4)

The twenty interviewees stated that they did not use UML

frequently. However, they acknowledged the various benefits of using it in software development. *"I understand that UML has a very strong semantic power, which favors its use in the elaboration of architecture, as well as in the construction of the system"* (P4). Störrle (2017) pointed out the importance of understanding the ever-changing demands of the software industry, which indicates more organizational and software development cultural differences as potential factors influencing UML use. Similarly, the results of Ozkaya and Erata (2020) show that the majority of professionals (88%) use UML in modeling their software systems from different architectural points of view. Among the architectural views (i.e., functional, information, concurrency, development, deployment, and operational), the most popular ones are functional and information views (96–99%). The operational point of view is the least popular, ignored by 61% of participants in their software modeling with UML. Studies (Kobryn, 2002; Dori, 2002; Thomas, 2004) argue that UML is not fulfilling the role of being a "lingua franca" or standard because of issues such as size, complexity, semantics, consistency, and model transformation.

Summary of RQ4: *The collected results show that UML modeling has low adherence in companies, although participants recognize the benefits of using UML models in software projects. These results are consistent with previous studies.*

4.6 RQ5: How does the context of software projects in companies limit the use of UML?

Figure 6 presents the collected data associated with RQ5. Three project context issues have been summarized that may affect UML use: (a) UML formalism (or lack thereof) – would more formalism in UML lead developers to use it more frequently? (b) the use of UML for practitioners arises from the fact of adapting its use for a specific purpose, and (c) companies tend to develop relatively small software that undergoes continuous modification. Participants indicated that the high demand for software development may end up limiting the use of UML in practice. Thus, developers start to keep design decisions "in mind" (or through informal communication channels) and communicate effectively without any formal diagram.

More formalism. Regarding UML formalism, Figure 6 (a) shows that 28% are neutral, 27% partially agree, and 21% totally agree that more formalism would help UML use. Of the 20 participants we interviewed, 15 consider that the high degree of formalism becomes a negative factor for the applicability of UML since the processes are highly dynamic and agile, requiring a less formal and more interactive use. The project context our interviewees were involved in is usually very dynamic and agile, thus leading to constant changes in design, documentation, and UML models when they exist. More formalism in the language may lead to higher effort in producing and maintaining up-to-date models in such dynamic and agile scenarios. Therefore, even though some participants seem to understand the benefits of having more

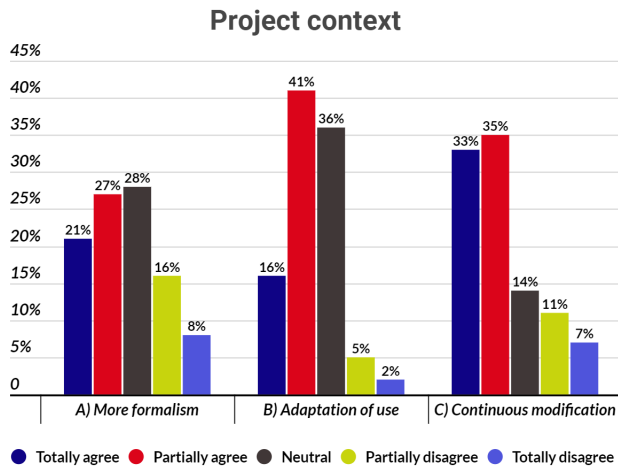


Figure 6. Context of use (RQ5)

formalism in modeling languages (e.g., more code generation and model transformations), in most of today's projects, there are not enough resources to take up the high cost of creating and maintaining semantically-rich models (with a higher degree of formalism).

Adaptation of use. Figure 6(b) summarizes to what extent participants agree that the UML use correlates to whether they can adapt it to their specific needs. The majority of the interviewees (12) pointed out that UML can be adaptable to a specific purpose (e.g., project domain, a specific section of the architecture, or a specific stakeholder's view), but this adaptation is complex due to factors such as 1) it costs a lot to assure that documents/models are in sync with the code; 2) the difficulty in measuring the return on investment of adopting modeling practices; 3) UML use in legacy software; 4) the fear of adopting changes in the process, especially when working with more mature teams that have grown without modeling practices. Therefore, that all leads us to believe that much research is still needed.

Continuous modification. Figure 6(c) summarizes data on whether participants agree that the continuous modification nature of relatively small to medium projects makes it difficult to use UML. That data also matches with interviewees' perceptions. Even when practitioners work on larger projects, they usually break them into smaller iterations (and sub-projects) where developers can get along without much modeling activity. Although the study participants of Petre (2014) believe, for the most part, that UML is a "lingua franca" in companies and that they have theoretical knowledge about this type of modeling, participants end up not using it frequently. The results of Fernández-Sáez et al. (2015) revealed that software developers using UML diagrams end up experiencing difficulties with reading them. Therefore, most surveyed companies use the "most understandable" UML diagrams. Maintainers do not always use the available documentation and work directly with the source code; even when documentation with models is available, it is not typically used.

Summary of RQ5: *The project context matters. Depending on the project and process, more or less formalism might help UML use. Also, the ability to continuously update diagrams together with continuously changing code in specific projects is another influencing factor. Finally, whether it is possible to adapt modeling practices to specific project needs affects UML use.*

4.7 RQ6: How do practitioners view UML modeling?

Figure 7 summarizes data regarding RQ6. We explored three possible issues related to practitioners' views on adopting UML modeling.

Not interested in modeling. Figure 7(a) shows that 41% totally agree, 33% partially agree, and 13% are neutral on whether they are interested in modeling tasks. Additionally, out of the 20 participants interviewed, 13 stressed that developers like and understand the importance of modeling; however, factors (discussed in RQ1 and RQ2) limit its adoption. In Petre (2013), UML is considered "unnecessarily complex" by several participants in that study who reported variations in understanding and interpretation among developers, resulting in problems such as challenges in formal language semantics. Others noted that the complexities of the notation limited its usefulness – or required targeted use – in discussions with stakeholders (including highly technical stakeholders).

Lack of modeling pattern. Figure 7(b) indicates that 15% fully agree, 37% partially agree with the lack of modeling patterns and modeling guidance; in other words, the open-ended nature of UML makes it less attractive. According to the interviewees, this lack of modeling guidance on creating models correctly and effectively prevents developers from using UML modeling. "Not all project participants will understand modeling, there is no pattern. There are no people qualified to generate UML" (P5). In Hutchinson et al. (2011b,a), they found that there are various modeling languages people use in projects following model-driven engineering (MDE). Companies using MDE tend to develop domain-specific languages (DSLs), which have a very product/implementation-focused notion.

General model. Figure 7(c) shows that 19% fully agree and 39% partially agree that the lack of a general diagram that provides a system big picture with structural and behavioral elements makes the UML adoption less attractive. Most of the interviewed participants (16) reinforced the difficulty of modeling structural and behavioral aspects of complex software in a single "big picture view."

Fernández-Sáez et al. (2018) sought to provide a comprehensive and systematic view of the main challenges in software modeling and to understand the different categories of them together with discussions of the concrete challenges in each category that professionals may face. In their study, they raised eight different types of challenges, including managing the complexity of the language, extensive modeling languages, domain-specific modeling environments, developing formal modeling languages, analyzing models, sepa-

ration of concerns, transforming models, and management models.

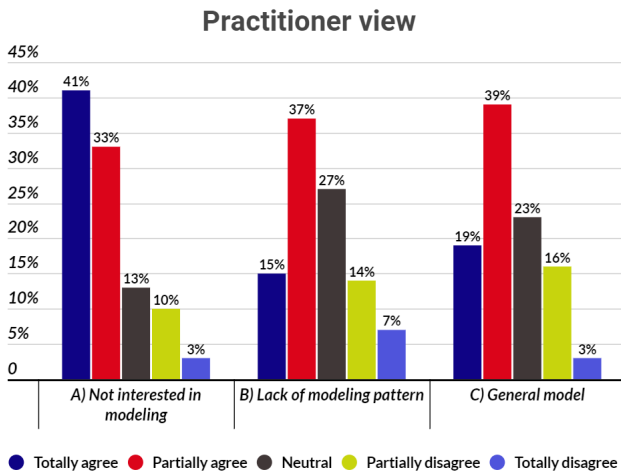


Figure 7. Practitioner view (RQ6)

Summary of RQ6: *Most developers do not demonstrate an interest in modeling, which can be justified by crucial factors such as the absence of standard modeling guidance and difficulty bringing upfront design aspects to the software development lifecycle. New modeling approaches are required to facilitate modeling and bring developers closer to it, making the process simpler, more dynamic, and motivating.*

5 Additional Discussion

In Section 5.1 we provide reflections and future directions based on the obtained results. Section 5.2 discusses issues related to the adoption of continuous modeling. Section 5.3 outlines some discussions on gamified software modeling as a way to enhance the adoption of UML models. Section 5.4 discusses the need for new approaches to assess UML diagrams in the context of modeling training education. Section 5.5 draws implications from our findings.

5.1 Summary of Reflections

Time constraints and lack of knowledge. The study results point to time constraints as one of the main factors that affect the use of UML. Although participants recognize the importance and benefits of creating UML diagrams, the short time spent on projects leads professionals not to use UML or to use it in a limited manner. In addition, the lack of in-depth knowledge about UML diagrams would be an impediment since the cost of promoting proper understanding among people with different levels of education/experience and viewpoints is high. Also, the ability to evaluate the UML model quality is another considerable challenge.

Academic vision. One factor that interviewees consistently pointed out was the impression that UML tends to be more academic than industrial/practical and that new teaching approaches need to be adopted in academic programs

that involve UML in their curriculum. Software engineering education training with UML needs to be accompanied by real problems from the industry, which reinforces the findings from Neto et al. (2021). It is evident that regardless UML is a dominant representation in practice, there is evidence that it plays an important role in software engineering teaching (Petre, 2014). UML provides a common representation from which to direct the system design discussion and build a shared model of the problem. It provides a means for “model-based thinking” for students who do not yet have a repertoire of representations and reasoning tools. The typical use of UML in education introduces key concepts and directs attention and structure to student exploration and practical involvement with problems and design. One can argue that the value of UML in education lies in intellectual development rather than mirroring industry practice.

Company culture and agility. From the responses we got, we identified that the culture of agility in companies conflicts with the use of UML. Preparing and maintaining UML diagrams are two manual activities requiring knowledge and time. Therefore, the popularity of informal modeling (e.g., whiteboard sketches) has grown as an attempt to improve collaboration and communication effectiveness. Also, informal and lower-cost models (in the sense of being more straightforward and faster to draw) become more flexible since learning is simplified. Usually, working with the representation of abstractions (i.e., modeling) in the context of agility culture has not proved to be a popular choice. Delivering quickly (without major planning) and considering failures as a natural process to arrive at the final software product have proved to be a priority. In this context, the multi-vision modeling proposed by UML does not find any application space, although it is recognized as something important.

Selective use of diagrams and complexity. When asked about what benefits they perceived when using UML, most participants responded that using UML diagrams selectively (i.e., the use of only a few diagrams) helps to minimize complexity, avoids problems of inconsistency between diagrams, and helps in forming a common understanding between developers. This conclusion was also verified by Dzidek et al. (2008). The generality and freedom that enable UML to meet this wide range of purposes are also the sources of its weakness. UML has no formal semantics, which poses a problem when people use an UML model for different purposes. Because one of UML’s main objectives is to communicate software design, different ways of using UML are potential causes of communication problems (Lange et al., 2006).

5.2 Adoption of continuous modeling

Companies seek not only to streamline their processes but mainly to find continuity throughout the software development cycle (Rubert and Farias, 2022; Chen, 2015; Elazhary et al., 2021; Chen, 2017; Laukkanen et al., 2017; Fitzgerald and Stol, 2017). Fitzgerald and Stol (2017) argue that achieving flow and continuity throughout the software development cycle is much more important in the first instance than velocity. Since companies increasingly prioritize continuous delivery practices (Chen, 2015), to benefit from UML adoption (Bucchiarone et al., 2021; Chaudron et al., 2012; Dzidek

et al., 2008), companies must put effort into involving UML modeling practices throughout the software development cycle. However, this requires significant process changes, for example, augmenting the CI/CD pipeline, giving rise to *continuous software modeling* (which poses a significant challenge).

Technical Challenges. Robust UML modeling approaches, tools, and good practices out-of-the-box and highly adaptable to the companies' realities are lacking. The absence of such an approach has led to the isolated, as opposed to continuous, adoption of UML models throughout the continuous delivery pipeline. Modeling tools that fill this gap can bring the already documented benefits of using UML models to the reality of companies, such as improving the traceability between models so as not to leave documentation and the process of modeling out of sync, not to mention the ability to highlight resource saving. When building a continuous modeling platform, different tools and technologies can be used as building blocks for the continuous delivery pipeline (Chen, 2015, 2017). However, companies should not be trapped by such tool suppliers. The scientific community should propose widely accepted modeling guidelines and good practices applicable to organizational needs companies typically experience, define open APIs (software modeling as a service) and build an ecosystem of tools for building a continuous software modeling pipeline.

Nowadays, software development iterations are short to of delivering newly requested features rapidly, establishing a continuous cycle of getting feedback. Such large, monolithic models need to be characterized and rethought as feature-oriented UML models. Modeling practices must fit iterative processes (with very short release cycles) that are typically driven by incremental feature development. Rather than designing a colossal set of UML diagrams upfront, it is recommended that software design with UML follows the same iterative approach driven by incremental feature development, which may ease the adoption of software modeling in agile teams. That poses the significant challenge of implementing continuous modeling approaches oriented by features, as well as the production of empirical evidence about the advantages and disadvantages of adopting continuous UML modeling. Solving this challenge will require close collaboration between researchers and practitioners and will enable the benefits of UML modeling to be brought to the reality of more companies.

Process challenge. Xavier et al. (2019) pointed out that people still associate UML modeling with traditional process practices (e.g. RUP), while UML is not explicitly integrated with agile practices. Our results indicate that agile teams tend not to adopt UML modeling. One of the participants reported: *"if the preparation of UML models requires, for example, three days before it is ready for use by developers, this period will be responsible for much of the sprint time, for example"* (P12). It is important to highlight that agile methodologies do not prohibit the use of UML, another participant states: *"we work with Scrum and with some UML diagrams, but few and only in the project phase. The system is giant to meet a bank's demands, there are many requests for functionality changes and improvements on the part of the customer and we usually*

fit the demands into weekly sprints" (P11). There are research gaps in looking for alternatives, aiming at the alignment between business processes, agile development practices, and UML modeling.

Documentation and legacy monolithic systems. Promoting large-system modeling practices without processes that support documentation is still a challenge for decades. There may also be the cultural tendency to assume that the status quo is the only possible path. The absence of design documentation complicates restructuring legacy monolithic systems into highly distributed systems such as those following the microservice architecture. Legacy systems typically have dozens of tightly coupled subsystems that interact to provide different services for internal and external customers within companies. Fitzgerald and Stol (2017) point out that the lack of documentation only takes into account the tacit knowledge of software engineers who work in different teams.

The legacy systems modeling based on creating a "big picture view" is still hard to implement due to the size, usually consisting of hundreds of thousands of lines of code. Continuous updates to these models can be very challenging. The multiview modeling of UML allows updating complementary models, such as class diagrams and sequence diagrams. This can lead to inconsistencies between such models (Kretschmer et al., 2021; Khelladi et al., 2019; Reder and Egyed, 2013).

5.3 Gamification of modeling software

Gamification can be defined as "the use of game design elements in non-game contexts" (Deterding et al., 2011; Huotari and Hamari, 2017; Liu et al., 2017). This technique uses the philosophy, elements, and mechanics of game design in non-game environments, aiming to bring all the positive aspects they provide. The current literature recognizes the benefits of applying gamification in software engineering practice. However, how to design and use gamification in the context of modeling applied to industrial needs is still an open question. As far as we know, only a few studies on the application of gamification in software engineering practices are available — most of which are related to broader contexts (Porto et al., 2020; Pedreira et al., 2015; Ren et al., 2020).

Due to the related theoretical and practical difficulties, learning to use the full potential of UML can be a complex task, which makes developers feel discouraged and less engaged over time. This scenario could lead, for example, to the development of incomplete, decontextualized, and poor-quality models. Lange et al. (2006) reinforce that this issue brings potential risks that might cause misinterpretation and miscommunication, thus reducing software quality. Therefore, finding configurations that favor developer practices, generate engagement, and consequently, increasingly effective UML models can become one of the main challenges encountered in the industry today. Given this scenario, gamification emerges as a possible alternative to mitigate these problems, enhancing the adoption of UML, improving the models generated by developers, and generating high-quality software.

There is no clear and usually accepted taxonomy of game elements (Pedreira et al., 2015). Shpakova et al. (2016) pro-

posed a unified view of the different classifications, which summarizes gamification in three dimensions: Components, Mechanics, and Dynamics. Components are the basic building blocks of gamification. They represent the objects that users see and interact with, such as badges, levels, and points. Mechanics define the game as a rules-based system, specifying how everything behaves and how the player can interact with the game. Dynamics are the top level of gamification elements. They include all aspects of the game that cannot be implemented and managed directly and are related to users' emotional responses (e.g., progression, exploration).

The success of gamifying a particular context unrelated to the game depends heavily on the gamification design choices for those three dimensions. Several research efforts have focused on identifying the phases that make up the gamification project (Mora et al., 2015; Webb, 2013). However, similarly to the taxonomy of gamification elements, there are no commonly accepted phases. They can vary in number and terminology used. In software development, developers' performance concerning productivity or quality may relate to the number of artifacts developers produce and how good the artifacts are. However, while performance is often a quantitative and objective metric for assessing the impact of gamification on users' activities in the out-of-game context, in software development, performance may be related to productivity or quality (usually subjective).

This article conjectures that the insertion of gamification techniques, such as feedback, progress, and challenges, in software modeling could help mitigate the issues of adopting UML modeling. For example, the incompleteness of UML models is a critical problem (Lange et al., 2006; Fernández-Sáez et al., 2018). Using gamification techniques, such as challenges, points, feedback, and progress, could motivate developers to create more complete models in exchange for points, for example. A ranking system for software teams could be created to rank them in terms of the quality of the models created. In addition, constant feedback during the model editing could foster learning and stimulate modeling. Researchers can carry out empirical studies to analyze the integration between gamification and software modeling based on the factors mentioned in RQ1 and RQ2. That would increase the perception of benefits by practitioners (RQ3) and the frequency of use (RQ4). Therefore, the use of gamification techniques can motivate developers, enhance the quality of the created UML models, and foster learning.

5.4 Assessing and Grading UML Diagrams

Before using UML models, practitioners need to learn that there are structural and behavioral diagrams available in UML. Also, students (or practitioners under training) submit their diagrams for assessment and grading in educational/training contexts. University courses worldwide teach UML modeling, to some extent, as the standard language for modeling software. Additionally, UML is still a well-known language when practitioners need to model software systems.

Moreover, universities are increasingly adopting a learning-by-doing approach and having online classes with a high number of students. In this context, students need to practice through hands-on exercises and real-world

tasks. Instructors must find an efficient mechanism to fairly and equitably assess student projects and assignments. In addition, assessments must enable rapid feedback and provide learners with instructions on how to overcome their deficiencies or limitations. Imagine that an instructor needs to train 120 people in geographically distributed teams. The instructor provides an exercise in which the learner needs to design 10 UML class diagrams. The instructor needs to provide feedback on the 1,200 UML class diagrams two days after delivery. The short time to evaluate a high number of diagrams makes the teaching and learning process difficult. Therefore, the manual assessment of UML models proves to be a very costly and subjective activity, creating friction in the practice-assessment-learning feedback loop involving students and instructors. This reality is not exclusively found in universities; on the contrary, it is found anywhere where the teaching-learning cycle of UML models needs to happen quickly and with a relatively high number of learners.

Some tools and approaches (Vesin et al., 2018; Bian et al., 2019; Stikkolorum et al., 2019) have been proposed in recent years. For example, SDMetrics⁶ presents a set of metrics for UML models but does not compute the differentiation between the rubric and the UML model created by the learner. The ModelGuru approach⁷ goes a little beyond SDMetrics when computing students' grades using object-oriented measures of design size, coupling, and complexity. Vesin et al. (2018) came up with a new integrated tool to support the evaluation of UML models produced by students. Bian et al. (2019) introduced a grading process based on syntactic, semantic, and structural matching for computing grades by comparing students' models with the desired model. In a different approach, Stikkolorum et al. (2019) presented an exploratory study regarding machine learning for grading UML diagrams.

However, a streamlined approach for grading UML diagrams based on syntactic, semantic, and structural criteria is still lacking. The use of machine learning also emerged as a trend and a new avenue to be explored. Lastly, we outline the need for the scientific community to explore three objectives (Farias and Silva, 2020): (1) provide a tool to streamline the process of managing rubrics for grading UML diagrams; (2) allow students to get faster and more objective and itemized feedback for their submissions; and (3) ultimately, enhance the practicing-grading-learning feedback loop associated with designing UML diagrams.

5.5 Practical Implication

When software development teams constantly change source code and revise UML models to keep them up-to-date, the effort engineers put in can make the difference between adopting or not UML models throughout the development process. From our findings, updating and synchronizing models with source code appears to be one of the major impediments to the broader use of UML modeling. Rather than being easy and intuitive, study participants point to model update and synchronization as a highly time-consuming and error-prone process.

⁶SDMetrics: <https://www.sdmetrics.com/>

⁷ModelGuru: <http://modelguru.snotra.com.br/>

Still, the need to update and synchronize UML models attracts the spotlight as organizations increasingly adopt DevOps and agile practices in globally distributed development teams. Therefore, updating and synchronizing (UpSync) UML models with source code emerges as a critical requirement to leverage UML adoption in real-world settings. The ability to “UpSync” UML models can be seen as the mitigation with which modern development teams (adept to DevOps and agile practices) can update the UML’s structural and behavioral models to accommodate new design decisions, or requirements change. We conjecture that the greater the UpSync, the better the quality of the source code. This paves the way for the scientific community to propose friendly round-trip engineering approaches — existing UML models can be transformed into source code and then be converted back — combined with the integrated development environment used by development teams.

In that perspective, updating and synchronizing models helps improve the software system under maintenance. Previous empirical studies (Dzidek et al., 2008) have shown that using UML models improves source code quality and reduces bugs. For this, not only robust round-trip engineering approaches are needed, but also improvements that span the agile development process as a whole. For example, SCRUM-based development processes can have automated tasks at the end of each sprint to update and synchronize UML models.

Practical Research Implication: Our findings highlight that the adoption of UML modeling in practice is affected by the difficulty of updating and synchronizing models with the source code. Currently, development processes adopt source code as a primary artifact, thus demanding that new cost-effective updating and synchronization approaches be proposed. Although UpSync models sound like a promising trend, the scientific community needs to evaluate future proposed techniques and carry out empirical studies to investigate the impact on the quality of UML models and source code, as well as on the degree of practitioners’ satisfaction in real-world settings.

6 Threats to validity

This section discusses the possible threats to the study’s validity.

Internal validity. Internal validity is related to issues that may affect the causal relationship between treatment and outcome. Threats to internal validity include instrumentation and selection threats. The main points affecting our study’s internal validity refer to the participants’ profiles and experiences. When analyzing the profile of the participants, as presented in Section 4.1, around 30% of them have low (up to 4 years) general experience, low experience with software modeling, and low experience with software development. This is probably because the level of education of about 50% of that 30% group is low compared to others, or they are still attending undergraduate degrees. In addition, many of these participants may not have studied UML yet during their undergraduate degrees. Also, there was no option in the sur-

vey question corresponding to the time of 2 to 3 years of experience. However, due to the sample size and also the complementary interviews we conducted, we believe that the data collected are not affected by this threat. Another internal threat is linked to the random process of selecting participants for the interview, which may have caused a potential similarity in the profile of the interviewed participants. Thus, a selection bias may interfere with the potential validity of completion. Although the interviewed participants work on software development in the fields of education, agribusiness, e-commerce, government, trade, product export, and finance, we recognize that qualitative data could be further explored if we had greater participation of professionals linked to other sectors. Still, we have a wider variety of sectors from the survey participants.

External validity. External validity concerns the ability to generalize the results beyond the actual study. To perform the correct interpretation of the survey results, although the demographic data of our sample are diversified, we understand that the generalization of them for the entire population may not be adequate. In our study, participants belonged to a geographic variety and worked in companies of different domains and sizes. However, we cannot be sure that this sample is representative of the sector in general. We understand that these threats are always present in industrial research. Reliability focuses on the replicability of results by other researchers. This study has a repository with the collected data and an online form, both of free access.

7 Conclusions and Future Work

This article presented an exploratory survey on how practitioners have used UML modeling in the Brazilian industry. In total, 376 employees from 210 information technology companies answered an online questionnaire about the factors affecting use, difficulty, and frequency of use, perceived benefits, and contextual factors that prevent the adoption of UML models. In addition, we interviewed 20 randomly chosen participants from the survey pool using a semi-structured interview protocol as a follow-up investigation to triangulate with the survey data.

In summary, the results show that: 74.8% of the participants answered that they do not use UML frequently. Participants who responded not to use UML models attributed factors such as continuous delivery practices, time constraints, lack of knowledge about modeling, company culture, and the always present difficulty of keeping the models up to date and synchronized with each other and source code.

The results of this research reinforced some evidence already found in the literature concerning the use of UML (Gorschek et al., 2014; Petre, 2014). In general, most people know UML but do not use it in their projects. These results can help professionals understand how to invest to avoid increased development spending and provide a foundation to motivate software developers to design UML diagrams throughout development cycles. That would facilitate, for example, maintenance tasks. Future work should focus effort on investigating more aspects related to UML practice, such as the possibilities of using UML in ag-

ile teams/organizations, whether teaching methodologies in academia influence the practices in the software industry, and how gamification can be applied to software modeling practices. Finally, we hope that the issues outlined throughout the article will encourage other researchers to replicate our study in the future in different circumstances and that this work represents a solid step in a more ambitious agenda to improve software engineering practices.

References

- Akdur, D., Demirörs, O., and Garousi, V. (2017). Characterizing the development and usage of diagrams in embedded software systems. In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 167–175. IEEE.
- Akdur, D., Say, B., and Demirörs, O. (2021). Modeling cultures of the embedded software industry: feedback from the field. *Software and Systems Modeling*, 20(2):447–467.
- Bian, W., Alam, O., and Kienzle, J. (2019). Automated grading of class diagrams. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 700–709. IEEE.
- Böhm, W., Junker, M., Vogelsang, A., Teufl, S., Pinger, R., and Rahn, K. (2014). A formal systems engineering approach in practice: An experience report. In *Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices*, pages 34–41.
- Bucchiarone, A., Ciccozzi, F., Lambers, L., Pierantonio, A., Tichy, M., Tisi, M., Wortmann, A., and Zaytsev, V. (2021). What is the future of modeling? *IEEE software*, 38(2):119–127.
- Chaudron, M. R., Heijstek, W., and Nugroho, A. (2012). How effective is uml modeling? *Software & Systems Modeling*, 11(4):571–580.
- Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50–54.
- Chen, L. (2017). Continuous delivery: overcoming adoption challenges. *Journal of Systems and Software*, 128:72–86.
- Cicchetti, A., Ciccozzi, F., and Carlson, J. (2016). Software evolution management: Industrial practices. In *ME@ MODELS*, pages 8–13. Citeseer.
- Ciccozzi, F., Malavolta, I., and Selic, B. (2019). Execution of uml models: a systematic review of research and practice. *Software & Systems Modeling*, 18(3):2313–2360.
- Deterding, S., Dixon, D., Khaled, R., and Nacke, L. (2011). From game design elements to gamefulness: defining “gamification”. In *15th Int. Academic MindTrek conference: Envisioning future media environments*, pages 9–15.
- Dori, D. (2002). Why significant uml change is unlikely. *Communications of the ACM*, 45(11):82–85.
- Dzidek, W. J., Arisholm, E., and Briand, L. C. (2008). A realistic empirical evaluation of the costs and benefits of uml in software maintenance. *IEEE Transactions on software engineering*, 34(3):407–432.
- Elazhary, O., Werner, C., Li, Z. S., Lowlind, D., Ernst, N. A., and Storey, M.-A. (2021). Uncovering the benefits and challenges of continuous integration practices. *IEEE Transactions on Software Engineering*.
- Farias, K., Garcia, A., Whittle, J., von Flach Garcia Chavez, C., and Lucena, C. (2015). Evaluating the effort of composing design models: a controlled experiment. *Software & Systems Modeling*, 14(4):1349–1365.
- Farias, K., Gonçalves, L., Bischoff, V., da Silva, B. C., Guimarães, E. T., and Nogle, J. (2018). On the uml use in the brazilian industry: A state of the practice survey (s). In *SEKE*, pages 372–371.
- Farias, K. and Silva, B. C. d. (2020). What’s the grade of your diagram? towards a streamlined approach for grading uml diagrams. In *23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–2.
- Fernández-Sáez, A. M., Caivano, D., Genero, M., and Chaudron, M. R. (2015). On the use of uml documentation in software maintenance: Results from a survey in industry. In *2015 ACM/IEEE 18th Int. Conf. on Model Driven Engineering Languages and Systems (MODELS)*, pages 292–301. IEEE.
- Fernández-Sáez, A. M., Chaudron, M. R., and Genero, M. (2018). An industrial case study on the use of uml in software maintenance and its perceived benefits and hurdles. *Empirical Software Engineering*, 23(6):3281–3345.
- Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.
- Forward, A. and Lethbridge, T. C. (2008). Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. In *Proceedings of the 2008 international workshop on Models in software engineering*, pages 27–32.
- Gorschek, T., Tempero, E., and Angelis, L. (2014). On the use of software design models in software development practice: An empirical investigation. *Journal of Systems and Software*, 95:176–193.
- Heldal, R., Pelliccione, P., Eliasson, U., Lantz, J., Derehag, J., and Whittle, J. (2016). Descriptive vs prescriptive models in industry. In *Proceedings of the acm/ieee 19th international conference on model driven engineering languages and systems*, pages 216–226.
- Ho-Quang, T., Hebig, R., Robles, G., Chaudron, M. R., and Fernandez, M. A. (2017). Practices and perceptions of uml use in open source projects. In *39th ICSE: Software Engineering in Practice Track*, pages 203–212. IEEE.
- Huotari, K. and Hamari, J. (2017). A definition for gamification: anchoring gamification in the service marketing literature. *Electronic Markets*, 27(1):21–31.
- Hutchinson, J., Rouncefield, M., and Whittle, J. (2011a). Model-driven engineering practices in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, pages 633–642.
- Hutchinson, J., Whittle, J., Rouncefield, M., and Kristofersen, S. (2011b). Empirical assessment of mde in industry. In *Proceedings of the 33rd international conference on software engineering*, pages 471–480.
- Jackson, D. (2019). Alloy: A language and tool for exploring software designs. *Commun. ACM*, 62(9):66–76.

- Júnior, E., Farias, K., and Silva, B. (2021). A survey on the use of uml in the brazilian industry. In *Brazilian Symposium on Software Engineering*, pages 275–284.
- Khelladi, D. E., Kretschmer, R., and Egyed, A. (2019). Detecting and exploring side effects when repairing model inconsistencies. In *12th ACM Int. Conf. on Software Language Engineering*, pages 113–126.
- Kitchenham, B. A. and Pfleeger, S. L. (2008). Personal opinion surveys. In *Guide to advanced empirical software engineering*, pages 63–92. Springer.
- Kobryn, C. (2002). Will uml 2.0 be agile or awkward? *Communications of the ACM*, 45(1):107–110.
- Kretschmer, R., Khelladi, D. E., Lopez-Herrejon, R. E., and Egyed, A. (2021). Consistent change propagation within models. *Software and Systems Modeling*, 20(2):539–555.
- Kuhn, A., Murphy, G. C., and Thompson, C. A. (2012). An exploratory study of forces and frictions affecting large-scale model-driven development. In *Int. Conf. on Model Driven Engineering Languages and Systems*, pages 352–367. Springer.
- Lange, C. F., Chaudron, M. R., and Muskens, J. (2006). In practice: Uml software architecture and design description. *IEEE Software*, 23(2):40–46.
- Laukkanen, E., Itkonen, J., and Lassenius, C. (2017). Problems, causes and solutions when adopting continuous delivery—a systematic literature review. *Information and Software Technology*, 82:55–79.
- Liebel, G., Marko, N., Tichy, M., Leitner, A., and Hansson, J. (2018). Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling*, 17(1):91–113.
- Liu, D., Santhanam, R., and Webster, J. (2017). Toward meaningful engagement: A framework for design and research of gamified information systems. *MIS quarterly*, 41(4).
- Mora, A., Riera, D., Gonzalez, C., and Arnedo-Moreno, J. (2015). A literature review of gamification design frameworks. In *2015 7th International Conference on Games and Virtual Worlds for Serious Applications (VS-Games)*, pages 1–8. IEEE.
- Neto, J. C., Bento, L. H. T. C., Oliveira Jr, E., and Souza, S. D. R. S. (2021). Are we teaching uml according to what it companies need? a survey on the são carlos-sp region. In *Anais do Simpósio Brasileiro de Educação em Computação*, pages 34–43. SBC.
- OMG (2017). Uml: Infrastructure specification. <https://www.omg.org/spec/UML/2.5.1/PDF>.
- Ozkaya, M. and Erata, F. (2020). A survey on the practical use of uml for different software architecture viewpoints. *Information and Software Technology*, 121:106275.
- Pedreira, O., García, F., Brisaboa, N., and Piattini, M. (2015). Gamification in software engineering—a systematic mapping. *Information and software technology*, 57:157–168.
- Petre, M. (2013). Uml in practice. In *2013 35th international conference on software engineering (icse)*, pages 722–731. IEEE.
- Petre, M. (2014). No shit or oh, shit!: responses to observations on the use of uml in professional practice. *Software & Systems Modeling*, 13(4):1225–1235.
- Porto, D., Jesus, G., Ferrari, F., and Fabbri, S. (2020). Initiatives and challenges of using gamification in software engineering: A systematic mapping. *arXiv preprint arXiv:2011.07115*.
- Reder, A. and Egyed, A. (2013). Determining the cause of a design model inconsistency. *IEEE Transac. on Software Engineering*, 39(11):1531–1548.
- Ren, W., Barrett, S., and Das, S. (2020). Toward gamification to software engineering and contribution of software engineer. In *4th Int. Conf. on Management Engineering, Software Engineering and Service Sciences*, pages 1–5.
- Rubert, M. and Farias, K. (2022). On the effects of continuous delivery on code quality: A case study in industry. *Computer Standards & Interfaces*, 81:103588.
- Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J., and Tortora, G. (2014). On the impact of uml analysis models on source-code comprehensibility and modifiability. *ACM TOSEM*, 23(2):1–26.
- Shpakova, A., Dörfler, V., and MacBryde, J. (2016). Gamification and innovation: a mutually beneficial union. In *BAM 2016: 30th Annual Conference of the British Academy of Management*.
- Stikkolorum, D. R., van der Putten, P., Sperandio, C., and Chaudron, M. (2019). Towards automated grading of uml class diagrams with machine learning. In *BNAIC/BENELEARN*.
- Störrle, H. (2017). How are conceptual models used in industrial software development? a descriptive survey. In *21st Int. Conf. on Evaluation and Assessment in Software Engineering*, pages 160–169.
- Thomas, D. (2004). Mda: Revenge of the modelers or uml utopia? *IEEE software*, 21(3):15–17.
- Usman, M., Felderer, M., Unterkalmsteiner, M., Klotins, E., Mendez, D., and Alégroth, E. (2020). Compliance requirements in large-scale software development: An industrial case study. In *Int. Conf. on Product-Focused Software Process Improvement*, pages 385–401. Springer.
- Vesin, B., Klačnja-Milićević, A., Mangaroska, K., Ivanović, M., Jolak, R., Stikkolorum, D., and Chaudron, M. (2018). Web-based educational ecosystem for automatization of teaching process and assessment of students. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, pages 1–9.
- Webb, E. N. (2013). Gamification: when it works, when it doesn't. In *International Conference of Design, User Experience, and Usability*, pages 608–614. Springer.
- Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.
- Xavier, A., Martins, F., Pimentel, R., and Carvalho, D. (2019). Aplicação da uml no contexto das metodologias ágeis. In *Anais do VI Encontro Nacional de Computação dos Institutos Federais*. SBC.