


Impact and Insights of MBE Adoption in Knowledge Capture and Organization: A Survey of Brazilian Software Companies

André Menolli  [Universidade Estadual do Norte do Paraná | menolli@uenp.edu.br]

Thiago Adriano Coleti  [Universidade Estadual do Norte do Paraná | thiago.coleti@uenp.edu.br]

Edson Oliveira Jr  [State University of Maringá | edson@din.uem.br]

Abstract Software development is a complex and knowledge-intensive process that involves multiple participants across various stages of the development lifecycle. Managing knowledge effectively in this context is challenging, especially when it comes to capturing, organizing, and reusing information in software engineering projects. Software development artifacts and techniques play a crucial role in addressing these challenges by facilitating the storage and sharing of knowledge. This study examines how model-based engineering facilitates various aspects of knowledge management. A survey of 62 Brazilian companies was conducted, providing a comprehensive roadmap for the future. Using qualitative and quantitative analyses, the findings were compared with those of other studies. The results indicate that different artifacts effectively support various knowledge management concepts across different phases of software development. Furthermore, while companies predominantly adopt artifact models in the early stages of development and recognize their benefits, they do not fully utilize their potential.

Keywords: *Models, software development company, Knowledge Management, roadmap, MDE, MBE*

1 Introduction

Software development is a knowledge-intensive process (Rus and Lindvall, 2002). It is common in software companies to have a dependency on critical employees, as much of their knowledge is tacit in nature. Consequently, it may jeopardize project issues in which such employees collaborate. Thus, companies need to manage such tacit knowledge to mitigate the potential problems (Rodríguez-Sánchez et al., 2020).

The concept of knowledge management (KM) aims to help companies minimize this kind of issue. KM is the process of creating, capturing, and using knowledge. Thus, it can be significantly transferred to another person (Ras et al., 2005). Therefore, it is essential to use KM techniques to systematically acquire, share, store, and reuse knowledge in the software development process (Vasanthapriyan et al., 2015).

The software development process encompasses activities from various existing fields. Additionally, software development employs approaches that utilize methods, processes, techniques, artifacts, and tools to design software. Thus, behavioral practices are increasingly important for software development and have gained much attention with the rise of iterative and flexible development methodologies (Flora, 2014). However, software development encompasses more than just adopting behavioral or managed process practices. Software development involves current knowledge of a given project and reuses knowledge from past projects. In addition, many companies do not adopt several important and efficient practices from development methodologies to share knowledge (Kobayashi et al., 2006).

Due to the complexity of the software development process, the adoption of behavioral practices is insufficient to manage knowledge effectively, particularly for large and distributed projects. In this sense, Ouriques et al. (2018) state that a significant amount of knowledge can be lost or not properly transferred to other individuals using informal com-

munication. Instead of disseminating the knowledge, it remains confined to the minds of a few individuals. Therefore, companies must use the correct software development techniques, tools, and artifacts to apply KM practices to acquire, store, and reuse knowledge (Menolli et al., 2015).

Although the importance of using artifacts, such as models, as organizational memory is evident, studies such as Bjørnson and Dingsøyr (2008) and Menolli et al. (2013), which explore how KM is related to software engineering, show the lack of studies to investigate the relevance of using models in KM. The term "model" is quite broad, making it challenging to provide a comprehensive yet precise definition. For our purposes, a model is an artifact created with the goal of capturing and sharing information in its purest form, minimizing any syntactical noise (Torchiano et al., 2013).

Some studies have shown that graph models provide better representations than textual models (Jolak et al., 2020, 2022) and help foster active discussions between stakeholders (Jolak et al., 2022). Furthermore, when models become a crucial and operational component of software development, the process is referred to as model-driven. Several model-driven techniques exist, including Model-Driven Engineering (MDE) (Schmidt, 2006), Model-Driven Development (MDD) (Mellor et al., 2003), and Model-Driven Architecture (MDA) (Kleppe et al., 2003). Regardless of the specific model-driven approach used, all are model-based. For this reason, in this study, we consider any model-driven approach as Model-Based Engineering (MBE).

Although extensive work has been done in the context of MBE, with numerous studies reported in the literature (Hutchinson et al., 2014; Jolak et al., 2020; Fernández-Sáez et al., 2018; Whittle et al., 2017; Torchiano et al., 2013), there is a lack of evidence regarding how MBE adoption impacts Knowledge Management (KM). This study aims to explore how MBE can support the knowledge management process. To achieve this, we focus on understanding the perceptions of

companies regarding how different techniques, methods, and artifacts used in MBE impact KM. We surveyed 62 Brazilian software development companies. Additionally, we mapped these methods, techniques, and artifacts, demonstrating their relationship with transformation processes in MDE and how they may aid in capturing and organizing knowledge. We provide a graphical roadmap called Model-based Engineering to Capture and Organize Knowledge (MBE4COK).

This paper enhances the understanding of how artifacts, particularly graphical models, and some requirement techniques and methods are used in MBE and how they relate to knowledge capture and organization throughout different stages of development. The MBE4COK roadmap offers insights into how the surveyed companies implement MBE and provides an overview of the main artifacts used in each type of MDE transformation. Finally, it presents a cross-comparison of our findings with previous MBE and MDE surveys.

This paper is organized as follows: Section 2 presents the theoretical background, which includes the main concepts used in this work and related works; Section 3 describes the research method used to conduct the survey; Section 4 presents the survey results; Section 5 analyzes results and provides a roadmap; Section 6 discusses the results presented; Section 7 discusses threats to validity; and the conclusion is presented in Section 8.

2 Background

This section presents the essential concepts of model-based approaches and discusses related work.

2.1 Model-Driven Engineering and Model-Based Engineering

In this section, we provide a brief overview of the modeling concepts involved in the survey, as there are numerous definitions of models, artifacts, model-driven engineering, model-based engineering, and model-driven development in the literature.

The term “model” is broad, making it challenging to establish a precise yet comprehensive definition. A model is an abstract representation of the system or its aspects, which ignores certain details and is created for specific purposes (Sommerville, 2011; France and Rumpe, 2007). In essence, a model is an artifact created to capture and communicate information as clearly as possible while minimizing syntactic interference. Common examples include UML design models, BPMN process models, Web application models defined using WebML, and textual models based on Domain-Specific Languages (DSLs) (Torchiano et al., 2013).

Due to the diversity of models and the variety of processes in which they are applied, setting clear boundaries for modeling is a complex task. Additionally, models can be utilized in different ways and at varying levels of maturity (Tomassetti et al., 2012).

When models serve as a fundamental and operational component of software development, the approach is known as model-driven development. Considering the context of

software modeling, different terminologies are presented in the literature. In our survey design, we addressed the two terminologies offered by Kolovos et al. (2006); Stahl and Völter (2006) to describe the differences between “model-based approaches” and “model-driven approaches”. While in Model-Driven Engineering (MDE), the models play a central and active role and “are at least as important as the source code” (Stahl and Völter, 2006), in the model-based approach, the models play an important role, although they are not necessarily the key artifacts of development (Brambilla et al., 2017). In our survey, we agreed that model-based does not “drive” the process as in MDE. Therefore, all model-driven processes are model-based, but not vice versa. Thus, this study focused on companies that use models following model-driven approaches and those that do not follow a model-driven approach, which can be described as Model-Based Engineering (MBE).

It is essential to distinguish MDE from Model-Driven Development (MDD). While MDE describes approaches to creating and using abstract models of applications transformed into concrete implementations (France and Rumpe, 2007), MDD is a paradigm that uses model artifacts in the development process (Stahl and Völter, 2006). A very common third term is Model-Driven Architecture (MDA), which is a particular vision of the Object Management Group (OMG) of the MDD and, thus, relies on the use of OMG standards (OMG, 2003). Therefore, considering the various nomenclatures involving software development or model-driven software engineering, in this work, we adopt MDE as it is a more generic term that covers any approach driven by models.

As our survey addresses the model-based approaches and the MDE, it is important to present the main MDE concepts involved. MDE technologies combine two concepts (Schmidt, 2006): Domain-specific modeling languages (DSML) and Transformation. DSML formalizes the application structure, behavior, and requirements within particular domains.

The transformation term has different understandings in a model-driven context. When this term is used in MDD or MDA, it refers to automatic transformations. It involves a set of transformation rules, which, together, describe how a model that uses a source language can be transformed into a model or text that uses a target language (Bennett et al., 2010). However, in this work, we adopted the more general concept of MDE. Transformation at this level means an operation that takes a set of artifacts as input and produces a collection of artifacts and a set of trace links as output (Kolovos et al., 2006). Therefore, we consider transformation in the model-driven context, any transformation that occurs from an input model and produces an output artifact, which can be a model or a text (a final code or intermediary code, for instance), manually, semi-automatically, or automatically. Thus, model-to-model (M2M) transformations and model-to-text (M2T) transformations are considered in the scope of this work.

The transformations enable refinement models to be developed to different levels of abstraction until they generate source code. Models at a higher level of abstraction are the Computation Independent Models (CIM) that represent the system independently of the computational solution to be de-

veloped and do not show the application's structure.

Following the abstraction level, the Platform Independent Models (PIM) represent a view of the application with no direct association with the platform's characteristics, making it possible to have a model suitable for different types of similar platforms (OMG, 2003). These models are abstracted from the technology and are used later in the development life cycle. On the other hand, platform-specific models (PSM) contain all the information required on the behavior and structure of the application on a specific platform.

Therefore, in this study, we analyzed the use of models at different stages of software development and considered whether the use of models followed a model-driven approach. Furthermore, we also studied the models' transformations during the development process.

Additionally, we employed various classification types to define the models. Such classification takes into consideration the stages of the software development process (Requirements and Design), the type of the artifact (Dynamic or Structural), and the level of the model according to MDE (CIM, PIM, and PSM).

2.2 Knowledge Management Concepts

Software development is an intensive, knowledge-driven activity in which much of the knowledge produced resides in the minds of employees. Effectively managing this knowledge is critical for sustaining competitiveness and fostering innovation. In knowledge-intensive industries, such as software development, key assets are not physical factories or equipment, but rather the accumulated knowledge embedded in people, processes, and technologies (Nonaka et al., 2000). However, these companies often face challenges in systematically capturing, organizing, disseminating, and applying both the knowledge they produce and the knowledge they use.

Knowledge Management (KM) addresses these challenges by providing processes, strategies, and technologies to manage organizational knowledge systematically. KM is broadly defined as the process of creating, capturing, sharing, and applying knowledge to improve individual and organizational performance (Ras et al., 2005; Landoli and Zollo, 2008). Its primary goal, according to Tiwana (2002), is to enable the timely and effective application of fragmented knowledge by integrating it into business processes. This involves managing not only explicit knowledge, codified in documents, systems, and repositories, but also tacit knowledge, which is embedded in individual experiences and skills (Davenport and Prusak, 1998).

Organizational knowledge encompasses various domains, including business processes, relationships between organizational units, customer insights, market trends, and technical expertise (Yanzer Cabral et al., 2014; Davenport and Prusak, 1998). However, KM as a discipline is inherently interdisciplinary and complex. It often sparks debates around its boundaries and the distinctions between data, information, knowledge, and wisdom Yanzer Cabral et al. (2014).

In general, the KM lifecycle comprises three core stages Dalkir (2005): (i) knowledge capture and/or creation, (ii) knowledge sharing and dissemination, and (iii) knowledge

acquisition and application. This continuous cycle ensures that knowledge is leveraged throughout the organization to support decision-making, learning, and process improvement.

To assess and guide the implementation of KM initiatives, several maturity models have been developed. For example, the Knowledge Management Maturity Model (KMMM) proposed by KPMG KPMG Consulting (2000) and the KM Capability Maturity Model from APQC American Productivity and Quality Center (2025) provide structured frameworks for evaluating an organization's knowledge management capabilities. These models typically describe the progression from informal, ad hoc knowledge-sharing practices to fully embedded KM systems that are tightly integrated with organizational processes and strategic goals. As Dalkir (2005) highlights, KM maturity models serve as diagnostic tools that help identify gaps, prioritize interventions, and guide the evolution of KM. Similarly, Kulkarni et al. (2006) argues that achieving higher levels of KM maturity is essential for aligning knowledge processes with business objectives and sustaining competitive advantage.

KM support requires the use of various technologies, such as document management systems, collaborative platforms, knowledge repositories, and semantic tools. However, organizations still face significant challenges, including capturing tacit knowledge, fostering a culture of knowledge sharing, and ensuring that KM initiatives are aligned with their strategic goals. Overcoming these barriers is crucial to realizing the full benefits of KM.

In this work, we focus on understanding how MBE may aid the process of capturing and organizing companies' knowledge. An intrinsic concept connected to KM is Organizational Learning (OL), which describes how organizations capture, store, share, and use knowledge. Organizational Learning (OL) is a key concept connected to KM, as it involves transforming experience into accessible knowledge that supports organizational objectives (Senge et al., 1994; Nevis et al., 1995). OL is an adaptive process influenced by past experiences and relies on the development of routines and organizational memory. As highlighted by Menolli et al. (2015), individual knowledge shaped by personal abilities, beliefs, and experiences is fundamental for building organizational knowledge. Therefore, OL complements KM by enabling the consolidation and effective use of knowledge within organizations.

2.3 Related Work

In the literature, several related works investigate aspects of UML modeling and model-driven approaches. Some studies have investigated the state-of-the-practice of model-driven techniques or MBE via opinion surveys or case studies. Table 1 summarizes each study.

Agner et al. (2013) studied UML modeling and model-driven approaches for designing embedded software in Brazil. Their survey provided evidence of the maturity of UML and the adoption of model-driven approaches. Another relevant study on MDE and embedded software is Akdur et al. (2018), which examines these practices in projects, exploring the extent, motivations, and methods of software

Table 1. Related MDE or Model-Based Engineering Surveys

Study	Year	Focus	Approach	Domain
(Agner et al., 2013)	2013	Use of Model-driven approaches in the embedded software development industry	MDD	Embedded Systems
(Hutchinson et al., 2011)	2011	Adoption and application of model-driven software development in the industry	MDE	General
(Whittle et al., 2014)	2014	MDE's success and failure factors	MDE	General
(Akdur et al., 2018)	2018	Why and how software modeling and MDE practices in software engineering projects are used	MDE	Embedded Systems
(Grossman et al., 2005)	2005	Analyzed the perceptiveness and perceived ease of use in the UML adoption	MBE	General
(Nugroho and Chaudron, 2008)	2008	Impact of UML modeling styles	MDD	General
(Forward and Lethbridge, 2008)	2008	Software modeling experiences	MDE	General
(Torchiano et al., 2011)	2011	Modeling languages, processes, and tools with MDE	MDE	General
(Gorschek et al., 2014)	2014	Use of software design models in software development	MDD/MDE	General
(Torchiano et al., 2013)	2013	Modeling languages, processes, and tools with MBE and the factors influencing their adoption	MBE	General
(Hutchinson et al., 2014)	2014	Identify key factors influencing the success or failure of MDE adoption	MDE	General
(Fernández-Sáez et al., 2015)	2015	Analyzing the adoption of UML in software maintenance	UML	General
(Whittle et al., 2017)	2017	To analyze the impact of tools on the adoption of MDE.	MDE	General
(Moreira et al., 2022)	2022	An overview of the advancements and challenges in Model-Driven Requirements Engineering	MDD	Requirements
(Liebel et al., 2018)	2018	To evaluate current practices and identify challenges in MBE	MBE	Embedded Systems
(Ameller et al., 2019)	2019	Explores how non-functional requirements (NFRs) are managed in MDD in industry	MDD	Requirements

modeling and MDE usage. The findings indicate that UML is the most widely used language, with modeling approaches ranging from informal sketches to formalized models. Finally, Liebel et al. (2018) evaluates the state of practice and identifies challenges in the embedded systems domain resulting from shortcomings in MBE.

Hutchinson et al. (2011) conducted an empirical study on MDE. Their findings highlight the class diagram as the most commonly used diagram in MDE and examine respondents' perceptions of its impact on productivity and maintainability. In a subsequent study (Hutchinson et al., 2014), they identify key factors influencing the success or failure of MDE adoption, emphasizing organizational, managerial, and social aspects over purely technical considerations.

In Whittle et al. (2014), the authors examine MDE practices, focusing on the factors contributing to its success or failure. Their findings indicate that while MDE is often marketed as a code-generation solution, its true benefits extend beyond that. Expanding on this research, Whittle et al. (2017) investigates the impact of tools on MDE adoption and practice within a broader organizational context. Through industry interviews, the study identifies and categorizes tooling challenges, ultimately proposing a taxonomy of tool-related considerations.

The survey in Forward and Lethbridge (2008) explored the impact of UML modeling styles. The results indicated that the use of UML had the most significant effect on productivity during the design, analysis, and implementation phases. Another study on UML, presented in Fernández-Sáez et al. (2015), investigates its adoption in software maintenance. This study examines the effectiveness, perceived benefits, and profiles of companies that utilize UML documentation, based on a survey of industry professionals.

The study in Torchiano et al. (2011) explored software modeling experiences, revealing that UML was the most widely used notation. In a subsequent study Torchiano et al.

(2013), the authors investigated the relevance of software modeling and MBE, focusing on their applications (processes, languages, and tools) and the factors influencing their adoption or rejection.

Gorschek et al. (2014) investigated the modeling languages, processes, and tools in Italian software companies with MDE. The results indicate that UML is the language most frequently applied, and models are used often. Considering those who use models, almost fifty percent reported generating code from models.

Moreover, some studies focus on specific parts of software development, such as the work of Moreira et al. (2022), which explores the use of MDD in software requirements. The work Moreira et al. (2022) summarizes advances in Model-Driven Requirements Engineering, highlighting how MDD techniques support requirements specification, traceability, and analysis, especially in complex domains, reinforcing the role of models in improving automation and consistency in requirements engineering. Another study focusing on software requirements is conducted by Ameller et al. (2019), which provides an empirical survey on how non-functional requirements (NFRs) are managed in MDD in industry. The results indicate that while MDD offers benefits such as productivity and maintainability, it provides limited native support for many NFRs, often requiring manual adaptations.

Considering the studies presented, although artifacts and models are a well-explored area of software engineering, none of them investigate how MBE impacts KM, particularly in terms of capturing and describing knowledge.

3 Research Method

To understand how MBE is applied in software development and how it assists in describing and understanding informa-

tion, we planned and conducted a web survey with a cross-sectional data collection Wang and Cheng (2020).

The research followed the workflow steps for creating and conducting a survey, as shown in Figure 1. This workflow is based on the checklists for preparing and carrying out surveys introduced by Ghazi et al. (2019); Molléri et al. (2020).

The following sections describe how each flow step was applied in conducting the research.

3.1 Objective and Study Plan

To define the objective of this study, we performed the workflow steps depicted in Figure 2.

This study is aimed at **characterizing** how MBE is applied by software companies, **with the goal of** understanding the impact that MBE has on the description and comprehension of information, **from the perspective of** software engineering researchers, **within the context of** Brazilian software development companies.

We defined three research questions to guide this study as follows:

- **RQ1. What artifacts, techniques, and methods do companies use in their software development processes?**
- **RQ2. What kind of model transformations occur in the companies' software development processes?**
- **RQ3. How does MBE impact the organization and understanding of information?**

3.2 Population

In Brazil, as of 2023, there were approximately 37,602 companies in the software and services sector ABES (2024). Although these companies accounted for only a fraction of the global market in terms of quantity, they contributed to 1.5% of the total revenue generated by the software industry worldwide (ABES, 2024). Among them, 9,062 were dedicated specifically to software development, distributed as shown in Table 2.

Table 2. Distribution of software development companies in Brazil (ABES, 2024)

Company Size	Rate	# Employees
Micro	47.1%	Less than 10 employees
Small	45.3%	From 10 to 99 employees
Medium	4.5%	From 100 to 500 employees
Large	3.1%	More than 500 employees

The identified target audience consists of individuals working in software development companies across Brazil, holding various software engineering roles, including requirement engineer, business analyst, software developer, and tester. Preferably, we try to recruit more senior respondents who are familiar with the entire development process. In the recruitment process, we used accidental non-probabilistic sampling, and we targeted subjects via our institutional contacts and professional social network sites such as LinkedIn.

3.3 Design Research and Validation

The research instrument was an online questionnaire. We followed the flow presented in Figure 3 and first identified the modeling concepts and definitions to define the scope of the survey.

The survey encompasses 31 questions in three sections as follows:

- **Section 1 - Characterization of the organization.** We defined 11 questions to identify the characteristics of the respondents and the company they represent. We used the collected data to describe the survey demographics (Section 4.1);
- **Section 2 – Artifacts and Techniques.** It consists of 14 questions, which aim to identify which techniques and artifacts are used in different development phases by companies and the frequency of their use. This section also presents data to understand how the transformation among models happens in the surveyed companies;
- **Section 3 - Description and Organization.** It comprises 6 questions to identify the companies' perceptions of how model-based development and MDE influence the organization and understanding of information.

The research team, composed of three PhD-level researchers in the field of Computer Science, conducted the initial validation of the research instrument. We applied content validation, which provides evidence of the construct validity of an assessment instrument. This process ensures that the instrument accurately reflects the intended construct and supports the validity of the data collected and the inferences drawn from it (Haynes et al., 1995).

The research team also performed the construct validation. Construct validity is the degree to which an assessment instrument measures the targeted construct (Haynes et al., 1995). After internal validations, an external Ph.D. with more than ten years of experience in software development conducted a pilot on the instrument. In the pilot, the participant analyzed the instrument, filling out an evaluation on content analysis, instrument structure, adequacy with the research questions, understanding of the questions, difficulty in answering, and response time.

Thereafter, a pre-test was applied to a software developer on a real team in a software development company. Thus, the final survey questionnaire consisting of thirty-one questions is shown in Table 3. For each question, the type of answers is also mentioned, e.g., a single answer from a list or a Likert scale. Due to space constraints, we do not present the entire survey in this paper, but it can be found in an online source¹.

3.4 Application of Terms, Recruitment and Response Management

Before starting recruitment, we created the terms of free consent, confidentiality, and agreement, which all respondents signed before their effective responses. We recruited people in three ways: via email, via the social network LinkedIn, and through direct messages.

¹https://1drv.ms/b/s!ApJUjKrTZF95a6_oIrHmVdFBIC0?e=SqTe5k

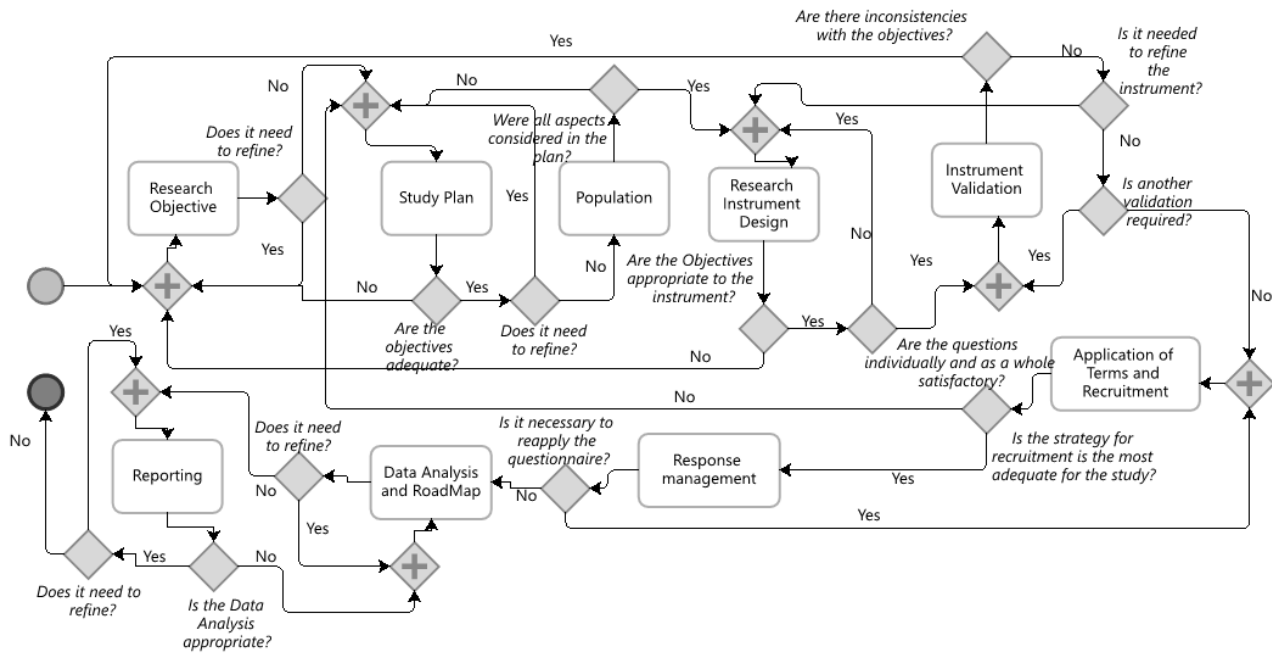


Figure 1. The performed workflow process for surveys

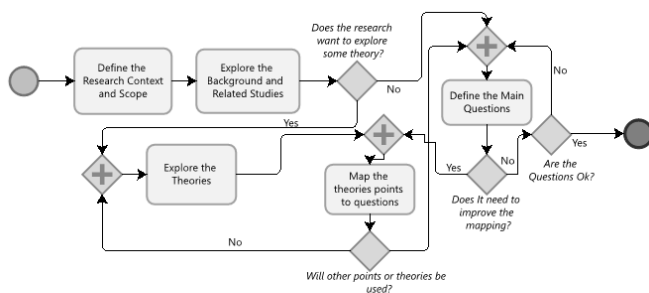


Figure 2. The Workflow Process for Research Objectives

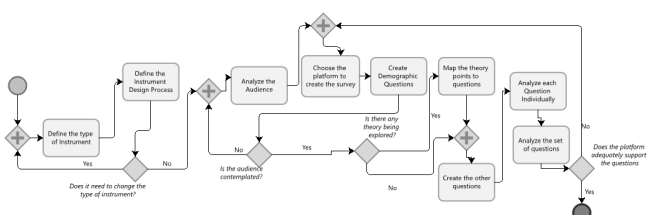


Figure 3. The Workflow Process for Research Instrument Design

After collecting the contact information, recruitment took place, and Table 4 summarizes the results. We can observe that the email response rate was low (5%). LinkedIn recruitment presented a response rate of 25%. An interesting data is that through LinkedIn recruitment, we obtained 90% of initial responses (the respondent started to answer the survey but did not finish it), against 28% of initial responses by recruiting via email.

Regarding the response rate, considering email recruitment, eight (21%) out of those who started answering the survey (38) have finished, while for LinkedIn recruitment, 54 out of 190 (28.4%) have finished. Finally, considering all the overall recruitment ($134 + 215 = 349$), the total response rate was 17.7% ($8 + 54 = 62$).

3.5 Data Analysis and Reporting

Figure 4 presents the data analysis process. We performed six activities in this process, which resulted in five outputs. First, we created the themes and topics of our analysis using Thematic Analysis (TA) and Content Analysis (Wohlin and Aurum, 2015). For the open questions, we used content analysis (Bardin, 1977).

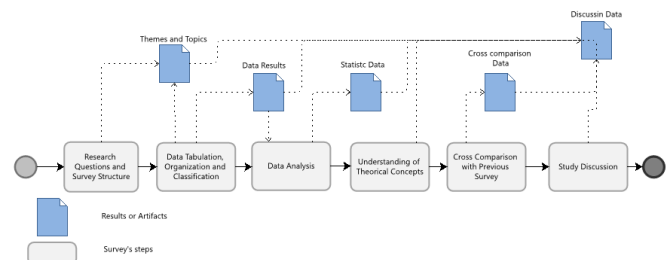


Figure 4. Data Analysis and Report Process

We have adapted the TA since it is generally applied to a set of texts. Instead, we used TA to identify common themes and topics in the questionnaire and data that could help answer our survey questions. We took as a starting point the work of Braun et al. (2014). Thus, we followed the six-phase approach to TA proposed by Braun and Clarke (2006). Afterward, the result data was generated using the organized and classified data. From the result data, we performed the data analysis (Section 5), in which we used three different strategies:

1. We extracted data regarding quantities, frequencies, and percentages and applied some statistical tests.
2. Using the data results and statistical analysis, it examines how modeling artifacts can enhance the organization and understanding of information.
3. We conducted a cross-comparison of our findings with previous MBE and MDE surveys. To plan the cross-

Table 3. Questions of the survey and respective types

Aspects	Questions	Type of Answers				
		Single answer	Multiple answers	Open-ended	Likert	Rel. Matrix
Characterization of the organization	Q1.State	X				
	Q2.City	X				
	Q3.Company Name			X		
	Q4.Year of Foundation of the Company			X		
	Q5.Start of IT activities			X		
	Q6.Research Respondent Position	X				
	Q7.The type of capital of the company	X				
	Q8.Company type	X				
	Q9.What countries do you serve?		X			
	Q10.What is the company's primary activity?	X				
	Q11.Company size	X				
Artifacts and Techniques	Q12.How is the interaction with the customer carried out in the phase of surveying the software requirements?		X			
	Q13. Is the person responsible for gathering software requirements the same person responsible for specifying the software requirements?	X				
	Q14.After the initial conversation with the customer, in what kind of document are the software requirements specifications stored?		X			
	Q15.What type of notation does the company use in specifying requirements?		X			
	Q16.Among the existing representations of software design artifacts, which ones are used to represent the structural (static) view in the company?		X			
	Q17.Among the existing representations of software design artifacts, which ones are used to represent the behavioral (dynamic) vision of the company?		X			
	Q18.When an artifact is used to generate another artifact or code, how is this accomplished?		X			
	Q19.In the software design phase, mark from which artifacts (columns) the structural (static) artifacts (rows) are created.					X
	Q20.In the software design phase, mark from which artifacts (columns) the behavioral (dynamic) artifacts (rows) are created.					X
	Q21.In which situations are models normally consulted by collaborators?		X			
	Q22.When changes occur to projects, what artifacts are changed to reflect the necessary changes?		X			
	Q23.How often and at what stages are the projects developed in the company represented by models?				X	
	Q24.The company's final codes (source codes, database tables, and graphical interfaces) are generated automatically or semi-automatically from which artifacts?					X
	Q25.What tools does the company use to transform models into source code (automatic or semi-automatic, total or partial)?			X		
Description and Organization	Q26. Evaluate the techniques below, considering how much each one favors the analyst's interaction with the customer to extract the initial requirements of the system?				X	
	Q27.Evaluate the forms of documentation below; how does each one favor understanding and description of the requirements?				X	
	Q28.Among the representations of the structural (static) view, evaluate how much each one favors the understanding and description of the information necessary for the later stages of development?				X	
	Q29.Among the behavioral vision (dynamic) representations, evaluate how much each one favors the understanding and description of the information necessary for the later stages of development?				X	
	Q30.Evaluate how much code generation (automatic or semi-automatic, total or partial) favors the items below				X	
	Q31.Please point out, according to your experience, what do you consider to be the main negative points or difficulties encountered in model-driven development?			X		

Table 4. The response rate for an invitation via email and LinkedIn

	Email				LinkedIn		
	Send	Opened	Started	Finished	Invited	Started	Finished
Total	134	54	38	8	215	190	54
Percentage	100%	40%	28%	5%	100%	88%	25%

comparisons, we itemized the types of findings reported by each of those studies and paired them (if any) with a question in our survey. Table 5 presents an overview of our plan for the cross-comparisons.

Finally, using data and concepts extracted from the literature, results, statistical data, and cross-comparison, we discuss the central questions of the work and provide a roadmap.

Table 5. Plan for cross-comparison of our findings with the previous surveys.

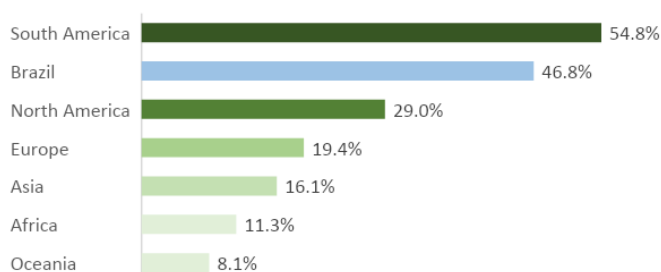
Previous Surveys	Points to be considered	Questions in our survey
(Agner et al., 2013) (Fernández-Sáez et al., 2015)	Use of models in Requirements	Q15
(Akdur et al., 2018) (Agner et al., 2013) (Fernández-Sáez et al., 2015) (Gorschek et al., 2014)	Use of Models in Design	Q16; Q17
(Akdur et al., 2018) (Whittle et al., 2014) (Agner et al., 2013) (Torchiano et al., 2013) (Hutchinson et al., 2014) (Whittle et al., 2017) (Liebel et al., 2018)	Benefits of model-based	Q30
(Torchiano et al., 2013) (Whittle et al., 2017) (Liebel et al., 2018)	Challenges of model-based	Q31
(Akdur et al., 2018) (Whittle et al., 2014) (Agner et al., 2013) (Torchiano et al., 2013) (Hutchinson et al., 2014)	Code Generation	Q24

4 Survey Results

We present the results in four parts. The first one is focused on demographics. The second part encompasses an overview of the techniques and artifacts. The third part aims to understand how the model transformation happened in the surveyed companies and to identify which input artifacts are used to generate output artifacts. The final section highlights the significance of the artifacts for various aspects of software development from the company's perspective.

4.1 Demographics

First, the profile of the companies surveyed is presented. Although the study was conducted with companies operating in Brazil, most companies operate in countries other than Brazil (Figure 5). Furthermore, 29% of companies are not Brazilian. Concerning the type of capital companies, 89% are private companies, 8.5% are mixed capital, and 3.5% are public companies. On average, the companies have been actively engaged in software development for over eighteen years.

**Figure 5.** Places where the surveyed companies operate

Regarding the company size, 48% are large, 26% are medium, and 26% are small or micro companies. Q10 was about the company's primary activity. Six possible choices were pre-provided in the questionnaire, and almost half of the surveyed companies develop custom software.

The current positions of respondents (Q6) are distinguished. Respondents have different roles in the companies,

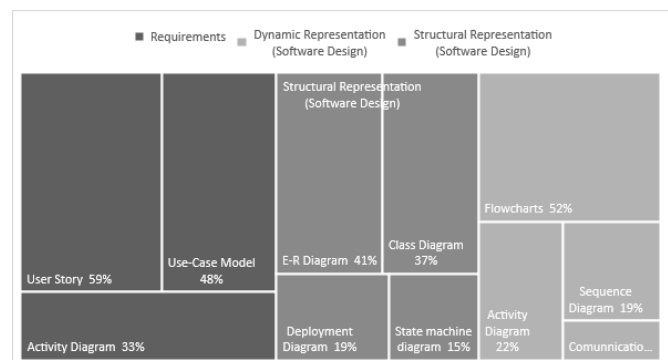
from “Chief Technology Officer” to “Software Tester”, and most of the participants have “System Engineering” roles.

It is important to note that, as it has been established in studies on information quality (for example, by Garvin (1988)), people in different positions see and rate the importance of other issues differently. Thus, they generally present varying viewpoints on SE and related processes.

4.2 Techniques and Artifacts Overview

We examined the artifacts and techniques employed during the requirements, analysis, and design phases in the surveyed companies. To provide a clearer understanding of the techniques and artifacts included within the scope of our survey, Table ?? presents a comprehensive list along with the corresponding development stages in which they were utilized.

Figure 6 summarizes the findings of Q15 and shows that the most frequently used artifacts in the requirement phase are the user story and use case model, while for the design phase, the most frequently used artifacts are the ER diagram, class diagram, and flowcharts.

**Figure 6.** Most used artifacts in requirements and design phases
Note: The question allowed multiple responses

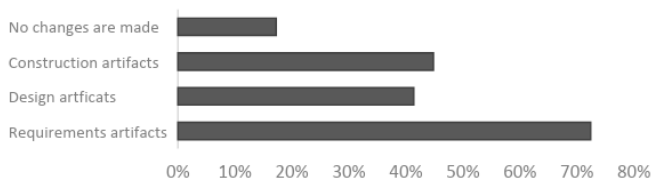
The following result presents the frequency with which surveyed companies use artifacts in different phases of development (Q23). We observed that the artifacts are most often used in the early stages. In the requirements phase, around

Table 6. Artifacts and Techniques Addressed in the Survey

Artifact or Technique	Source	Type	Phases Analyzed in the Study
Activity Diagram (AD)	UML	Graph Model	Requirement Specification, Analysis and Design
Architecture Description Languages (ADLs)		Specification Language	Analysis and Design
Business Process Model and Notation (BPMN)	OMG	Graph Model	Requirement Elicitation and Requirement Specification
Brainstorming		Technique	Requirement Elicitation
Business System Planning (BSP)	IBM	Method	Requirement Elicitation
Class Diagram (CD)	UML	Graph Model	Analysis and Design
Communication Diagram (COD)	UML	Graph Model	Analysis and Design
Deployment Diagram (DD)	UML	Graph Model	Analysis and Design
Eclipse Modeling Framework (EMF)	Eclipse Foundation (Chen, 1977)	Modeling Framework	Analysis and Design
Entity Relationship Diagram (ER)		Graph Model	Analysis and Design
Flowcharts (FLX)		Graph Model	Analysis and Design
Formal Method (FM)		Formal Notation	Requirement Specification
Formal Representation Language		Formal Notation	Analysis and Design
Formal Specification Language (FSL)		Formal Notation	Analysis and Design
Interface Description Languages (IDL)		Semi-Formal Language	Analysis and Design
Interview		Technique	Requirement Elicitation
Joint Application Design (JAD)	IBM (Darimont et al., 1997)	Technique	Requirement Elicitation
Knowledge Acquisition in Automated Specification (KAOS)		Method	Requirement Elicitation
Observation		Technique	Requirement Elicitation
Prototyping		Technique	Requirement Elicitation
Pseudocode and Program Design Languages (PDL)		Semi-Formal Language	Analysis and Design
Questionnaire		Technique	Requirement Elicitation
Sequence Diagram (SD)	UML	Graph Model	Analysis and Design
State Machine Diagram (SMD)	UML	Graph Model	Requirement Specification, Analysis and Design
Storyboard		Technique	Requirement Elicitation
Structure Chart		Graph Model	Analysis and Design
Use-Case Model (UC)		Graph Model	Requirement Specification
User Story (US)	UML	Visual Representation	Requirement Specification
Viewpoint Oriented Requirements Definition (VORD)	Kent Beck (Kotonya and merville, 1996)	Method	Requirement Elicitation
Workshop		Technique	Requirement Elicitation

24% of surveyed companies rarely use or never use models, while in the design phase, this number increases to 28%, and in the construction phase, it reaches 34%.

Q22 brings the artifacts modified when changes occur in the project. Figure 7 summarizes the answers and shows that around 77% of surveyed companies change requirements artifacts to conform to new demands and about 44% change project artifacts. On the other hand, 10% of companies claim they do not modify any artifact.

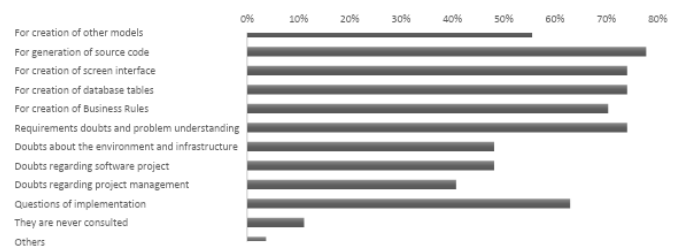
**Figure 7.** Frequency of changes in artifacts according to the software development phase

Note: The question allowed multiple responses

Figure 8 presents an overview of how employees access artifacts (Q21). Considering the results shown in Figure 8, we realized that the artifacts are most frequently consulted to generate final code (source code, interface, database, and business rules) and to solve requirements doubts.

4.3 Transformation Among Models

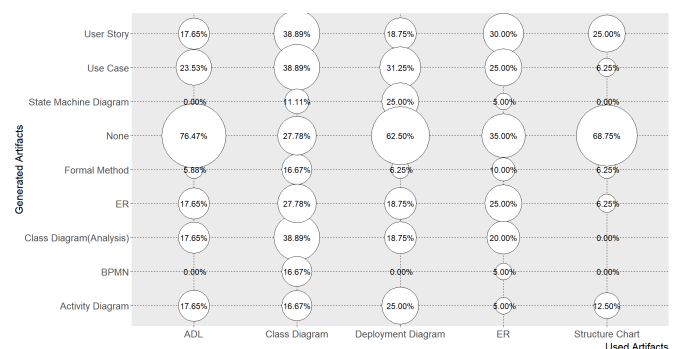
As exposed in Section 2.1, we consider transformations in the context of this work as some type of transformation, manual, semi-automatic, or automatic. Additionally, this research

**Figure 8.** Situations in which artifacts are accessed by employees

Note: The question allowed multiple responses

also considers the scope of both M2M and M2T transformations.

Considering Q18, it was identified that 80% of the surveyed companies perform transformation in some part of software development. Considering the most used artifacts, presented in Figure 6, Figure 9 shows structural artifacts and which artifacts are used in their generation (Q19).

**Figure 9.** Artifacts used to create structural artifacts

Analyzing Figure 9, it is noticed that most company-created artifacts are not transformed from other artifacts. On the other hand, around 50% of the companies reported using other artifacts to design class and E-R diagrams.

The most common artifacts as a base to create class diagrams and E-R diagrams are user stories. In addition, 28% of the companies (considering only companies that use class diagrams) reported using the E-R diagram as the basis for creating the class diagram, and 20% went the opposite (using class to create E-R).

Figure 10 shows the dynamic artifacts and which artifacts are used in their generation (Q20). For dynamic artifacts, considering only the artifacts used by the companies, we noticed more than 80% of them do not use any artifact to produce pseudocode. Furthermore, in companies that use sequence diagrams, 75% reported not using any artifact to create it. State machine diagrams also present a high rate of companies that reported not using any artifact to create it. This is somewhat surprising, as these diagrams aim to show the interaction of some objects to describe the behavior of some system functionality. On the other hand, more than 60% of companies use some artifact to create flowcharts. The most used are use case diagrams and class diagrams.

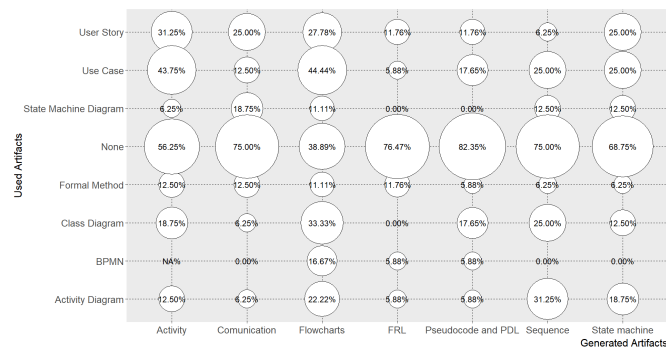


Figure 10. Artifacts used to create dynamic artifacts

Figures 9 and 10 consider all the transformations, not distinguishing M2M and M2T. Regarding M2T transformations, considering only companies that perform some type of transformation, Figure 11 presents the most common input artifacts to generate different output text artifacts (Q24). It is worth noting that class diagrams are commonly used to generate most output artifacts, except graphical interfaces, which are typically created from screen prototypes.

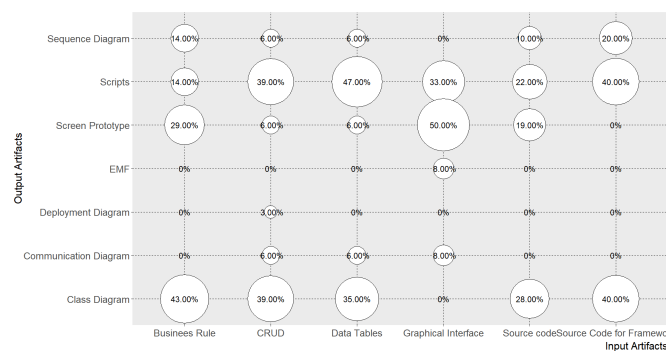


Figure 11. Input artifacts to generate output text artifacts

Only 33% of the companies reported automatic or semi-automatic transformations, and 47% reported that the trans-

formations occurred manually.

4.4 Importance of Models

Q26 presents data on how much each requirement technique supports the interaction between the analyst and the stakeholder to extract the initial requirements of the system. Analyzing Table 7, we observe that brainstorming, interview, and prototyping are techniques that better support the interaction with stakeholders to gather requirements, followed by a questionnaire.

Table 7. Importance of techniques to support interaction with stakeholders for gathering.

Artifacts	Support	Medium support	Not support	NGO*
Brainstorming	66%	10%	10%	14%
Interview	66%	14%	3%	17%
Prototyping	66%	7%	3%	24%
Questionnaire	59%	10%	17%	14%
Storyboards	55%	10%	7%	28%
Workshop	31%	17%	10%	41%

*The respondent did not give an opinion

Note: Question allowed multiple responses

The following results present the artifacts considered most essential to support the description and understanding of information necessary for the requirements and later stages of development. Table 8 summarizes the results of Q27, Q28, and Q29, presenting the mean score and standard deviation of each artifact, calculated from the individual notes attributed in the Likert scale, and the best scores are highlighted in gray. The artifacts best for describing information are the ER diagram, Flowchart, Use Case Model, and KAOS. The best artifacts to understand information are Use Case Model, KAOS, ER, Flowchart, and Communication Diagram. However, certain artifacts, such as KAOS, were reported not to have been used by many participants, which may have affected the results.

To better understand the relationship between artifacts and their importance in describing and understanding the information, multiple correspondence analysis (MCA) was used. The MCA is the generalization of Correspondence Analysis (CA) to several categorical variables (Greenacre, 2010).

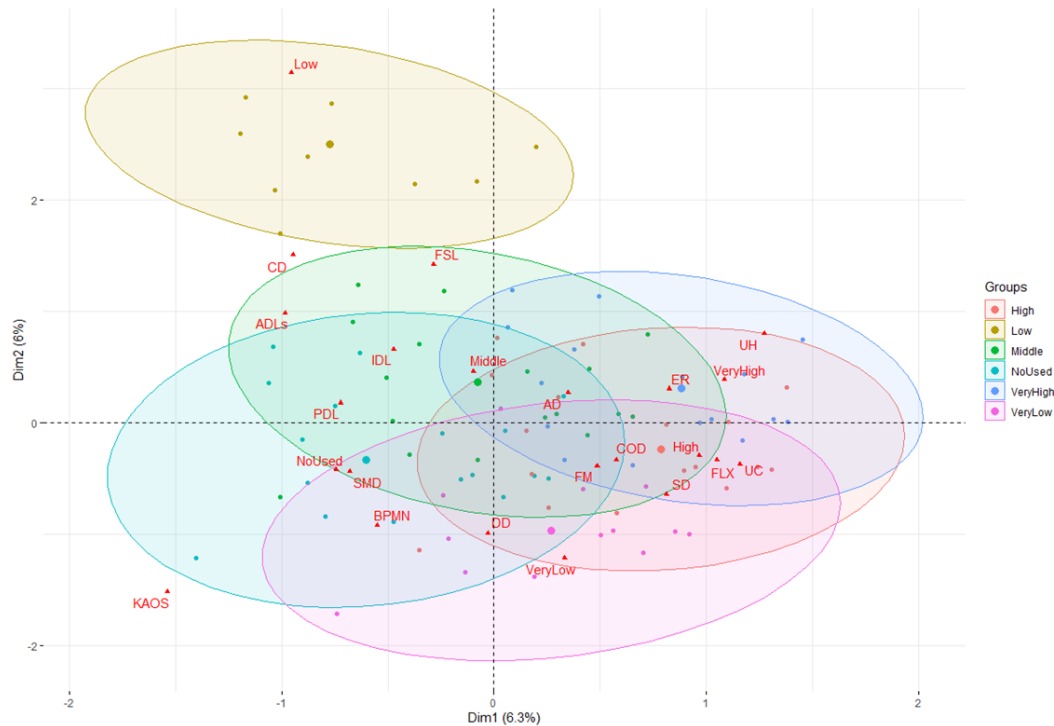
Figure 12 presents the MCA of the artifacts and their importance in describing the information. In this analysis, we used the answers to Q27, Q28, and Q29, which pertained to the extent of information provided about each artifact assigned by each respondent. Thus, applying MCA, groups (or rates of importance) were created, clustering the artifacts within each one. Figure 13 presents the MCA of the artifacts and their significance in understanding the information.

Observing Figure 12, the best artifacts to describe the information are ER, User Story, Use Case, Flowcharts, Sequence Diagram, and Communication Diagram. Figure 13 presents the best artifacts for understanding the information: ER, User Story, Use Case, Flowcharts, Sequence Diagram, and Class Diagram.

Furthermore, we used Hierarchical Clustering on Principal Component (HCPC) (Josse et al., 2010) to cluster artifacts according to their importance in describing and understand-

Table 8. Mean Score of importance and standard deviation for describing and understanding information, considering each artifact.

Artifacts	Understand		Describe	
	Score Mean	Stand. Dev.	Score Mean	Stand. Dev.
Activity Diagram (AD)	3.6	1.1	3.8	1.1
Architecture Description Languages (ADLs)	3.4	0.8	3.5	1.2
Business Process Model and Notation (BPMN)	3.9	1.1	3.7	1.2
Class Diagram (CD)	4.1	1.0	3.3	1.1
Communication Diagram (COD)	3.6	0.9	4.0	1.1
Deployment Diagram (DD)	3.8	1.0	3.8	1.1
Entity Relationship diagram (ER)	4.3	0.8	3.9	1.1
Flowcharts (FLX)	4.1	0.8	4.2	1.1
Formal Method (FM)	2.9	1.3	3.7	1.0
Formal Specification Language (FSL)	3.4	1.2	3.3	1.2
Interface Description Languages (IDL)	3.7	1.3	3.8	1.2
Knowledge Acquisition in automated Specification (KAOS)	4.0	0.7	3.2	1.3
Pseudocode and Program Design Languages (PDL)	3.5	1.0	3.4	1.3
Sequence Diagram (SD)	3.8	1.2	3.8	1.2
State Machine Diagram (SMD)	3.3	1.1	3.4	1.1
Use-Case Model (UC)	4.1	0.9	4.0	1.0
User Story (US)	3.6	1.3	4.1	1.0

**Figure 12.** Multiple correspondence analysis (MCA) of artifacts and their importance to describe information.

ing information considering the participants' perceptions as shown in Figure 14.

Finally, considering the importance of artifacts, it was analyzed whether the category of artifact (behavioral or structural) and the type of artifact (requirement or not) influence the description or understanding of the information. In general, we did not find an influence of the category type on the artifact to describe and understand information, as presented in Table 9.

The last result presents the perception of companies on how code generation may help essential aspects of software development (Q30). Figure 15 shows companies' perceptions of four axes.

Table 9. Mean Score of importance and standard deviation for describing and understanding information, considering categories and types of artifacts.

			Understand		Describe	
			Score Mean	Stand. Dev.	Score Mean	Stand. Dev.
Artifact Category	Behavioral		3.6	1.1	3.7	1.1
	Structural		3.9	0.9	3.8	1.1
Requirement Artifact	No		3.7	1.0	3.7	1.1
	Yes		3.5	1.1	3.7	1.0

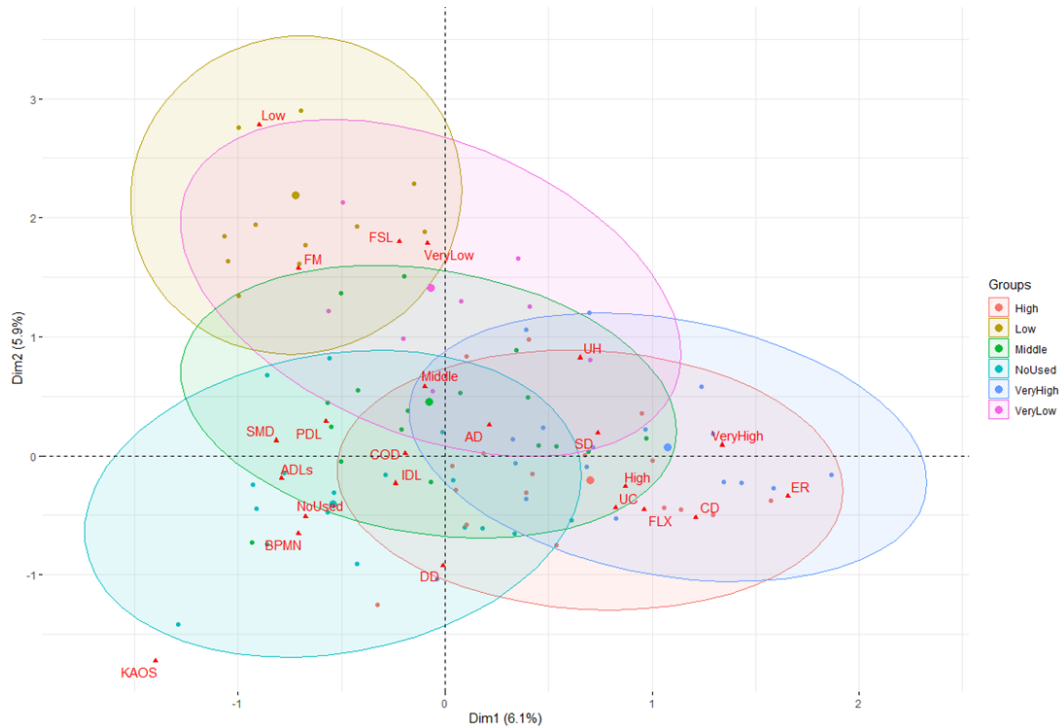


Figure 13. Multiple correspondence analysis (MCA) of artifacts and their importance to understanding the information.

5 Data Analysis and Roadmap

We used the themes and topics identified through TA to conduct the analysis. In this research, we adopted a deductive approach to TA, as we coded and developed themes based on predefined concepts or ideas. At the same time, we approached it in a constructionist way, as it focuses on understanding how MBE can help promote a better description and understanding of information.

The research instrument was analyzed to generate concise labels that identify the essential features of the data relevant to answering the research questions. After identifying themes, we analyzed the codes and collated data to uncover significant patterns of meaning. The candidate themes were then reviewed and validated against the dataset to determine the most critical ones for addressing the research questions. This process led to the generation of the final labels.

Finally, we identified the questions necessary to analyze each proposed theme in the instrument research. In Table 10, we present the research questions, the related themes, and the survey items explored in each theme. In the following sections, we offer the results for each theme.

5.1 Themes of the First Research Question

In this section, we present the analysis of the themes necessary to answer the first research question. To facilitate the analysis process, we created questions for the presented themes and answered them as follows.

For the theme **Artifacts/Technique** most used, we defined the question, “*What artifacts/techniques are most commonly used in each phase (Requirements/Design)?*”.

Ninety-six percent of surveyed companies rely on techniques to gather requirements. According to Figure 16, the most common forms of interaction with the stakeholders are

Table 10. Themes for each research question.

Research questions	Themes	Survey questions Analyzed
RQ1. What artifacts and techniques are used by the company in the software development process?	RQ1.T1. Artifacts/Technique most used	Q12; Q14; Q15; Q16; Q17; Q31
	RQ1.T2. Responsible for artifacts	Q13; Q21; Q22
	RQ1.T3. Frequency of use	Q23
	RQ1.T4. Code generation	Q24
RQ2. In soft. dev. what transformations between models occur?	RQ2.T1. Transformation among models	Q14; Q15; Q18; Q19; Q20; Q24
	RQ2.T2. Code generation	Q18; Q25; Q30
RQ3. How does MBE impact the organization and understanding of information?	RQ3.T1. Description and Organization	Q27; Q28; Q29
	RQ3.T2. Understanding	Q21; Q27; Q28; Q29

Interviews (77%), Brainstorming (74%), and Questionnaires (55%). More than 93% of the survey companies claim to store the gathered requirements records in some way, like in documents, tables, wikis, artifacts, or another type, and 55% of the companies use diagrams to represent them. Furthermore, the most commonly used notations for specifying requirements are User Stories (59%), Use Cases (48%), and Activity Diagrams (33%). In the survey Agner et al. (2013), 79% of companies reported using Use Cases and 47% using Activity Diagrams, while Fernández-Sáez et al. (2015) indicated that 39% of companies use Use Cases and 19% use Activity Diagrams.

In the design phase, more than 30% of the surveyed companies do not use any artifact for the structural representation of the software design, and more than 37% do not use any ar-

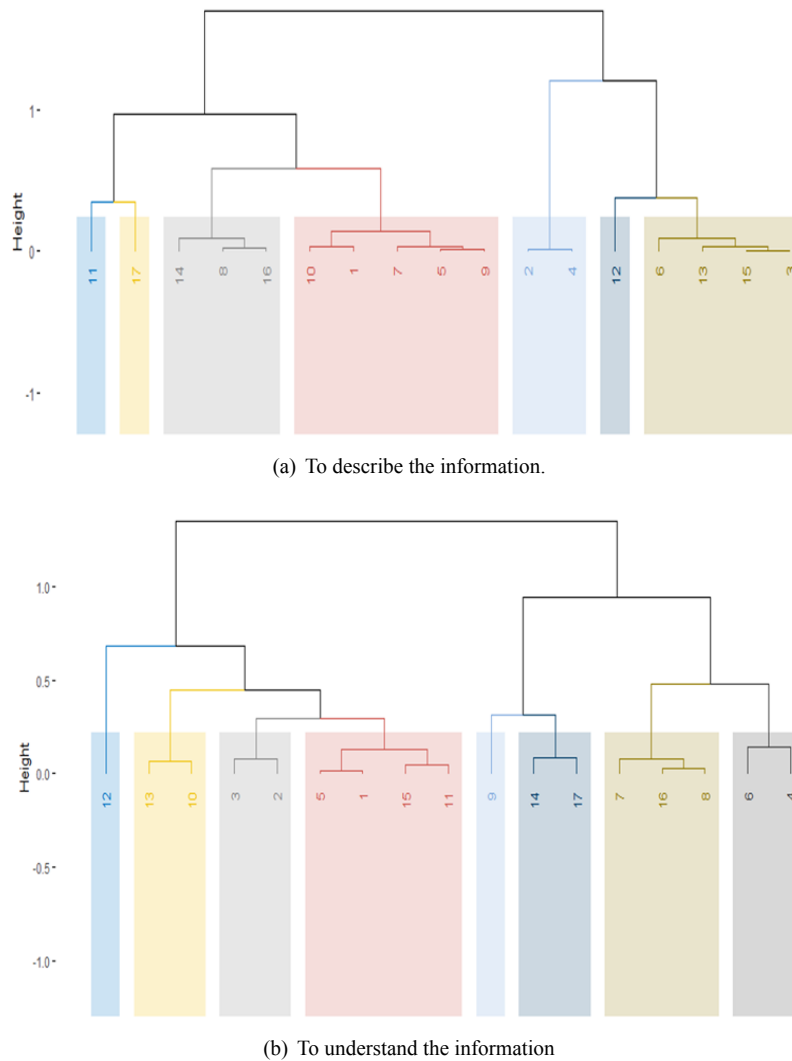


Figure 14. Artifacts clustered considering the levels of importance attributed: AD-1; ADLS-2; BPMN-3; CD-4; COD-5; DD-6; ER-7; FLX-8; FM-9; FSL-10; IDL-11; KAOS-12; PDL-13; SD-14; SMD-15; UC-16; US-17.

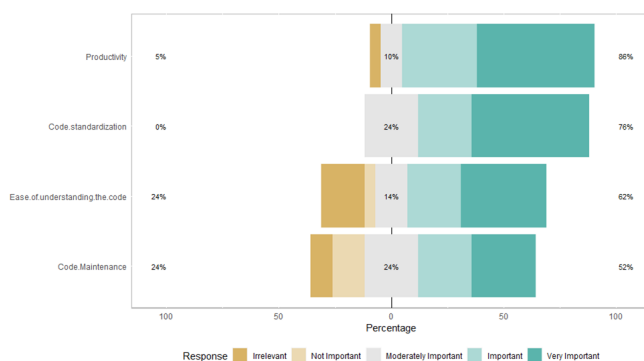


Figure 15. How the code generation (automatic or semi-automatic, total or partial) supports software development aspects.

tifact for the behavioral (or dynamic) representation. This is inferior to that found in Gorschek et al. (2014), where 48% of respondents do not use or rarely use design models. For structural representation, we highlight the E-R Diagram (40%) and the Class Diagram (37%), while for behavior representation, the artifacts more used are Flowcharts (51%), Activity Diagrams (22%), and Sequence Diagrams (18%). The results are closer to those presented by Akdur et al. (2018), which focused on responses to the embedded software indus-

try. More than 77% of respondents in the Akdur et al. (2018) study use the UML model, and Sequence Diagrams and State-machines Charts are the most popular diagrams, followed by Class and Activity Diagrams. In Agner et al. (2013), 99% of respondents use Class Diagrams, 94% Sequence, 27% Communication, and 91% State diagrams. Meanwhile, study Fernández-Sáez et al. (2015) reports that 39% of respondents use Class Diagrams and 29% use Sequence Diagrams.

Additionally, we identified the techniques used to generate requirements artifacts. Figure 16 shows such relations. A user story is typically created when interviews, prototyping, and brainstorming are employed. Activity and Use-case Diagrams are created when interviews, questionnaires, brainstorming sessions, and prototyping are employed.

We also compared the companies' perceptions of the best requirements, gathering techniques, and their use. We observe that brainstorming and interviews are considered the most effective methods to support requirements gathering. In contrast, storyboarding and prototyping are underutilized, despite receiving a superior rating in terms of their perceived ability to aid in extracting requirements, despite their limited use.

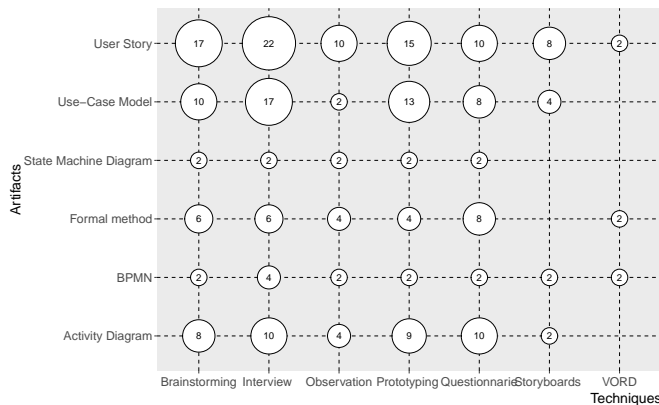


Figure 16. Requirement techniques used to create requirement artifacts

We also asked whether the artifacts are changed when there are modifications in the projects, as presented in Figure 7. Most surveyed companies change their requirements documents to conform to new demands, especially the requirements artifacts.

The main advantages of using artifact models are increased productivity, code standardization, and improved code comprehension. These findings align with previous studies. For instance, Agner et al. (2013) highlights productivity improvements in various aspects. Similarly, Whittle et al. (2014), Hutchinson et al. (2014), Torchiano et al. (2013), and Whittle et al. (2017) report that models enhance productivity. Akdur et al. (2018) further identifies key benefits, including cost savings, shorter development time, reliability, reusability, and maintainability. Regarding code standardization and comprehension, Torchiano et al. (2013) emphasizes the importance of quality and standardization, while Liebel et al. (2018) highlights quality, maintainability, traceability, and reusability as primary advantages of using models.

The main disadvantage is maintaining updated artifacts, which are rarely used in prospective activities. Companies report that modifying the code directly is more practical since updating artifacts requires extra time, increasing effort, and cost. These findings align with previous studies. For instance, Torchiano et al. (2013) identifies effort as a key challenge, while Whittle et al. (2017) highlights complexity and cost. Additionally, a common barrier to MDE adoption is the lack of supporting tools or their high cost (Torchiano et al., 2013; Whittle et al., 2017; Liebel et al., 2018).

The second question, related to the first theme, aims to answer: "Who is responsible for creating the artifacts and if these artifacts are really consulted in case of doubt?". In the requirements phase, in 66% of the surveyed companies, the same person who gathers the requirements is also responsible for its specification. Considering the size of the companies, 100% in micro companies, 87% in small-size companies, 44% in medium-size companies, and 53% in large-size companies, the same person who gathers the requirements is also responsible for its specification. Such models, as shown in Figure 5, were consulted in different situations, but more often, when there are doubts about the requirements and understanding of the problem, and for the generation of source code, interfaces, database entities, and business rules.

Continuing the data analysis to understand how artifacts are used by companies, the next question defined was "How

often and at what phases are projects developed in the company represented by models?".

The models are generally used most frequently in the early stages of software development. During the requirements phase, nearly a quarter of companies rarely or never use models. This number reaches 34% in the construction phase.

The last theme, tied to the first research question, addresses the generation of code and specifically tries to answer the question, "Can the use of artifacts aid in automatically or semi-automatically generating final codes?".

The companies reported using different artifacts to generate various kinds of final codes. The Class Diagram is used by 22% of the companies to generate source code and 19% to generate CRUD (Create, Read, Update, and Delete) code. Scripts are used by 23% of companies to create database entities and prototype screens are used by 19% of companies to develop graphical interfaces. The artifacts, as mentioned earlier, are primarily structural in nature. Behavioral artifacts are rarely utilized for generating final code.

Related surveys provide inconclusive evidence on the application of MDE in this context. Akdur et al. (2018) reports that 76% of surveyed companies use MDE for code generation, while Agner et al. (2013) states that 77% do not. Torchiano et al. (2013) found that model production is widespread (68% of respondents), but only 48% of modeling adopters use MDE techniques, with code generation being the most common. According to Hutchinson et al. (2014), 62% of respondents use executable models or simulations, and over 70% employ model-to-model transformations. Although MDE is often sold as a code generation solution; however, its real benefits don't necessarily lie in code generation (Whittle et al., 2014).

5.2 Themes of the Second Research Question

In this section, we present the analysis of the themes necessary to answer the second research question.

For the theme Transformation among models, we defined the question, "How do transformations between models occur, and which artifacts are used in these transformations in different phases of the software development?".

We need to examine the data from various perspectives to answer the question. The first is to understand which artifacts are used in the transformations. We observed that most companies transform between models at some stage of development. In the study Akdur et al. (2018), it was reported that 61% of researched companies use MDE-specific purposes (which include code generation, test-case generation, documentation generation, and M2M); in the study, Agner et al. (2013), the number was 23%.

The structural artifacts are mainly not created from other artifacts. The exceptions are Class Diagrams and E-R Diagrams; the artifacts most used to make them are use-case diagrams and user stories. It is essential to note that, although class diagrams are more expressive than E-R diagrams, they are more commonly used in the surveyed companies, with companies creating class diagrams from E-R diagrams more often than the other way around.

Most dynamic artifacts are not created from other artifacts. For instance, 75% of companies reported creating Sequence

Diagrams by not using any different model type. The Activity Diagram is the most commonly made from other artifacts. Almost 50% of companies reported using further diagrams as a base to generate Activity Diagrams, and the most cited is the Use-case Diagram followed by user stories.

Regarding M2T transformations, Class Diagrams are used most to create text outputs. The exceptions are data entity code, which is generated mostly from scripts, and GUI, which is mainly generated from screen prototypes. Table 11 summarizes the findings regarding the type of models (CIM, PIM, PSM, and text) and how the transformations occur among them. Considering the artifacts used in the transformations, it was not possible to make cross-comparisons with other surveys once none presented data at this level.

Only 12% of transformations are M2T, corroborating with the data presented in the results and data analysis section, which reports that most transformations occur manually. The most common transformation for structural diagrams is CIM to PIM, as Class Diagrams or E-R Diagrams are created from Use-case Diagrams or Activity Diagrams. CIM to CIM transformation is the most common for creating dynamic diagrams. It occurs because activity diagrams are used in different steps of software development. For instance, it is used in the design phase and can be created from a BPMN, flowchart, or use-case diagrams. The most common transformation is CIM to PIM, showing that transformations occur in the early stages of development.

Table 11. Types of Transformations

Transformation	Output Diagram		
	Structural Diagrams	Dynamic Diagrams	Total
CIM to CIM	6%	38%	20%
CIM to PIM	42%	32%	37%
CIM to text	6%	6%	6%
PIM to PIM	0%	6%	3%
PIM to PSM	37%	14%	27%
PIM to text	3%	1%	2%
PSM to text	6%	4%	5%
M2M	86%	88%	88%
M2T	14%	12%	12%

The second theme analyzed related to this research question was Code Generation. For this theme, we defined the question, “How does code generation from models occur?”.

We identified that code generation occurs manually in most parts of the company. Furthermore, as presented in Table 11, most transformations aim to create new models, rather than new code. Although most surveyed companies are not adopting code generation from models, they believe this practice may aid them in productivity, code standardization, code understandability, and maintenance improvement, as observed in Figure 15 and related in Gorschek et al. (2014), Santos et al. (2011), Torchiano et al. (2013), Liebel et al. (2018) and Whittle et al. (2014).

5.3 Themes of the Third Research Question

In this section, we present the analysis of the themes necessary to answer the third research question, taking into account the data provided in previous sections. The first analysis presents the perception of each kind of artifact and how it

may help describe software issues. Through Figure 12, which presents the MCA analysis, we observed that ER, User Story, Flowcharts, Sequence, and Communication Diagram are the best models for describing information.

Regarding the respondents’ perception of the best artifacts to help understand software issues, Figure 13 highlights ER diagrams, User Stories, Flowcharts, and Sequence and Class Diagrams as the most helpful models for comprehending information.

Although the Sequence Diagram is considered one of the best artifacts for describing and understanding information, and the Communication Diagram excels in describing information, these diagrams are less often used, as shown in Figure 6. Generally, the most widely used models are considered the most appropriate for describing and understanding information.

A final analysis regarding the best artifacts to describe and understand information is shown in Figure 14. It is observed that the deployment diagram was clustered together with the class diagram and the use case model, along with the ER, to understand the information. This clustering suggests that these models adhere to the same importance level standard, indicating that they can help understand the information. To describe information, the use case model was clustered together with the sequence diagram, and the formal method was clustered together with the communication diagram and ER diagram, suggesting that these artifacts help describe information.

Considering the purpose of understanding artifacts, Figure 8 shows that models are consulted throughout all phases of software development, especially when there are doubts about the requirements, understanding the problem, addressing implementation questions, and generating source code, interfaces, database entities, and business rules. Among specific techniques for requirements gathering, the surveyed companies identified interviews, brainstorming, and prototyping as the most effective for understanding and capturing knowledge. Additionally, storyboarding and questionnaires can also aid in this process.

5.4 The MBE4COK Roadmap

In this section, we present a roadmap named Model-based Engineering to Capture and Organize Knowledge (MBE4COK), depicted in Figure 17, aiming to understand the survey performed and its analysis. We believe that utilizing diverse artifacts and software development techniques is essential for capturing and organizing knowledge effectively. Consequently, for the effective use of software artifacts, companies must understand which artifacts can aid in organizing information and how to apply them.

The roadmap is divided into four frames: two about artifacts, “Artifacts and Techniques” and “Artifacts most used”, one about “Transformations”, and one about “Capture and Organize Knowledge”. In addition, the roadmap presents the relationship between artifacts, transformations, and the promotion of knowledge capture and organization.

The leftmost superior frame (“Artifacts and Techniques”) shows the main artifacts and techniques identified in the survey. The Requirements phase uses Activity Diagram

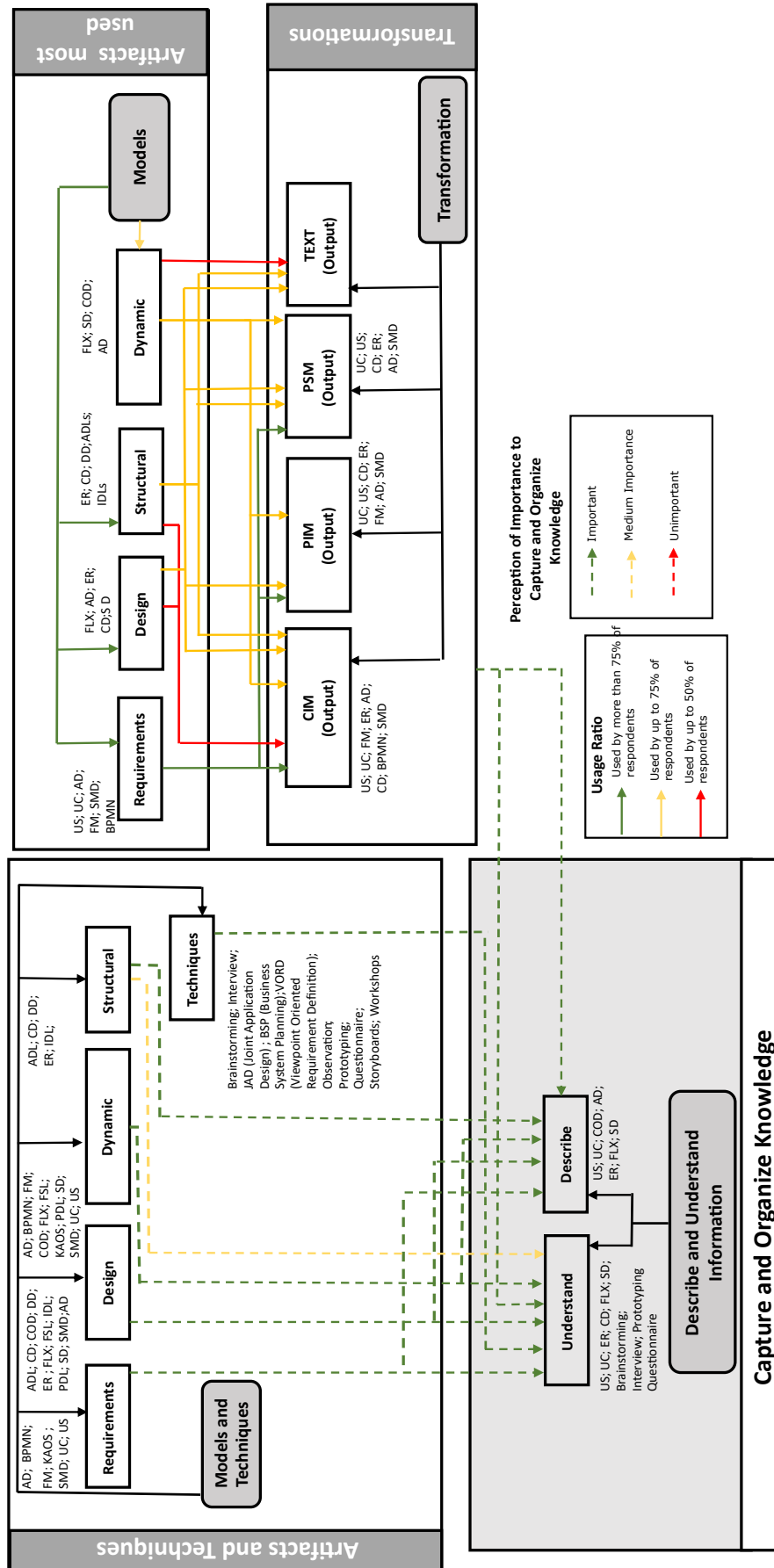


Figure 17. The MBE4COK Roadmap

(AD), BPMN diagrams, Formal Method (FM), KAOS, SMD, Use Cases (UC), and User Stories (US). In the Design phase, the following artifacts are used: ADL, CD, COD, DD, ER, FLX, FSL, IDL, PDL, SD, SMD, and AD. Dynamic models include AD, BPMN, FM, COD, FLX, FSL, KAOS, PDL, SD, SMD, UC, and US. Structural models are used as ADL, CD, DD, ER, and IDL. The following techniques have been used for companies' OL: Brainstorming, Interview, Joint Application Design (JAD), Business System Planning (BSP), VORD, Observation, Prototyping, Questionnaire, Storyboards and Workshops. Note that a colored dashed line highlights these artifacts and techniques, representing their perceived role in promoting knowledge, understanding, and organization. For instance, Requirements artifacts are crucial for knowledge understanding and description, while Structural artifacts hold moderate importance for understanding.

The rightmost superior frame ("Artifacts most used") shows the artifacts most used by the surveyed companies, according to different categories. It is possible to observe that all categories of artifacts are used. The text beside each category describes the artifacts most used. Requirements uses US, UC, AD, FM, SMD, and BPMN, whereas Design uses FLX, AD, ER, CD, and SD. Structural models include ER, CD, DD, ADL, and IDL. Dynamic models encompass FLX, SD, COD, and AD.

The frame "Transformations" presents each category of models and the type of output (CIM, PIM, PSM, and Text) generated, considering data from Figure 16 and Table 11. It is possible to realize that the models (CIM, PIM, and PSM) are frequently created from requirement artifacts, such as US and UC. Regarding CIM, design and structural models are not often used to generate them. Regarding code text, they are not frequently generated in surveyed companies. When generated from artifacts, structural models are generally used.

The "Capture and Organize Knowledge" frame shows surveyed companies' perceptions of how artifacts, techniques, and transformation aid in helping to describe and understand the information, using for this the analysis of the Figures 12 13 14 and Tables 9 8 7. From the results of the survey, the best artifacts for these two factors are US, UC, Flowcharts, ER, CO, and SD diagrams. In the majority, these artifacts fit the requirements, design, and dynamic. Thus, we can state that requirements, design, and dynamic artifacts are the most important in describing and understanding information. Moreover, it was realized that transformation, mainly through manual means, fosters the capture and organization of knowledge, as it is necessary to understand the information and thoroughly rewrite or reorganize it.

5.5 How to Surf the MBE4COK Roadmap

The artifacts are used for various purposes by diverse stakeholders in the context of software development (Störrle, 2017). For instance, models are used as communication and collaboration mechanisms when there is a need to solve problems and achieve a common understanding of the overall design within a group (Gorschek et al., 2014). Other surveys conducted (Akdur et al., 2018; Agner et al., 2013; Whittle et al., 2014) present various reasons for using models, includ-

ing maintainability and productivity. Therefore, the industry often belittles the use of artifact models to manage knowledge.

The software development process is knowledge-intensive, and teams depend on their collaborators. Furthermore, the turnover in software development teams is increasing. Thus, in this scenario, teams should focus on enhancing knowledge capture and organization, which can be partially achieved through the use of proper techniques and artifacts. Therefore, the MBE4COK aims to support the selection of artifacts, methods, and transformations in the software development process.

To select the appropriate artifacts and techniques, it is necessary to identify the stage of development and the types of knowledge that need to be managed. For example, if we are going to start a new requirement with a stakeholder, we should:

- First, use techniques and artifacts to extract the information. The best artifacts for this are BPMN, Use Cases (US), and User Stories (UC). In many cases, building these artifacts requires socializing with stakeholders, and for this purpose, interviews, prototyping, and questionnaires are helpful.
- Once the requirement artifacts are externalized, we need to consider how easily information can be extracted from them. Once again, US, UC, and flowchart are the most appropriate.
- Finally, we need to consider how to make the knowledge accessible to the entire organization. The US, UC, and flowchart artifacts are the most suitable; workshops, interviews, and brainstorming techniques may aid in the process.

The MBE4COK is also useful in determining the type of document to use for creating transformations. Let's suppose that some companies want to create artifacts from other artifacts:

- At the CIM level, usually, the artifacts are created from requirement artifacts, such as US, UC, and BPMN.
- At the PIM level, requirement artifacts are the most used. However, structural artifacts, such as ER and class diagrams, are also commonly used.
- At the PSM level, once again, requirement artifacts are mainly used to help generate PSM. Structural and dynamic artifacts can also aid in creating diagrams, such as entity-relationship (ER) diagrams, class diagrams, and sequence diagrams.
- Finally, at the text level, which is more associated with automatic transformations, structural artifacts used for design mainly help create the final code.

6 Discussion

This study examines the impact of MBE on the capture and organization of knowledge. To achieve the primary goal of this study, three questions were asked and discussed as follows. Considering the presented results, data analysis, and

roadmap, it was feasible to answer the main research questions proposed in this study.

Considering the first question, “What artifacts, techniques, and tools do companies use in their software development processes?”, the most used technique to gather requirements is the interview (77%), followed by brainstorming (74%). The notations most used to specify requirements are user stories (59%) and use cases (48%). In the design phase, the artifacts most used are flowcharts (51%), E-R diagrams (40%), and class diagrams (37%). We observed that models are most commonly used in the early phases of software development. In addition, companies generally modify such models in the early stages as well.

The second question is, “What kind of model transformations occur in the companies’ software development processes?”. Model transformation is a standard process in surveyed companies but is rarely used. Considering the classification adopted in this study and presented in Section 2.1, the most common type of transformation occurs among models (M2M) (88%) and is also manually applied. Regarding the type of artifacts, we observed that it is more common to create structural artifacts than dynamic artifacts in model transformations. About the kind of models, the most common transformation occurs between CIM and PIM, followed by PIM and PSM. Considering M2T, generating text from CIM models is more common than PSM. We also observed that traceability is not the primary reason companies use transformations, as the artifacts produced from transformations are generally not utilized in subsequent transformations during later stages of development.

The findings in our survey align closely with those in other studies regarding transformation and code generation. In the study of Yanzer Cabral et al. (2014), most participants in the survey reported that they did not conduct model-based automatic code and document generation. The survey Akdur et al. (2018) identified that traceability was not one of the most critical factors in MDE use, and Hutchinson et al. (2011) states that one of the negative influences of the MDE is the effort required to develop new transformations or customize existing ones. Therefore, model-driven approaches like MDE are commonly sold as transformations and code-generation solutions. However, its real benefits are not necessarily in these concepts (Whittle et al., 2014).

The third question of this study is “How does MBE impact the organization and understanding of information?”.

Using artifacts in the software development process, whether an MDE or model-based approach, aids in improving knowledge management. For instance, Gorschek et al. (2014) found in a survey they conducted that modeling in software development is primarily used as a communication and collaboration mechanism when there is a need to solve problems and achieve a common understanding of the overall design within a group.

Our study found that although most companies do not frequently adopt automatic transformations in their processes, they believe it may help improve certain aspects of software development, such as productivity, code standardization, code understandability, and maintenance.

Regarding code standardization, using artifacts aids in a common language and the externalization of knowledge at

a high level of abstraction. Productivity can be achieved in various aspects, including the time required to develop code, test code, model, or maintain it. The study Akdur et al. (2018) states that using the model increases individual and team productivity, and Whittle et al. (2014) describes how companies surveyed in their research seem to experience productivity increases of between 20% and 30%. Personal productivity is enhanced by a clearer understanding of the systems, facilitated by a high level of abstraction, and team productivity is improved by better communication and models that make knowledge accessible to the company.

Furthermore, Hutchinson et al. (2011); Akdur et al. (2018), Hutchinson et al. (2014) identify that MDE positively impacts maintenance. Using models enables a better understanding among stakeholders, as they make it easier for new staff to comprehend existing systems (Hutchinson et al., 2011).

We interpret the data provided by surveys of the literature and by our survey, and based on our result analysis, we bring the following scenery:

- Describe the Information: In different phases of software development, various artifacts can assist in this task.
- Understand the Information: This task is more challenging to achieve than just describing it. However, it can be achieved through the systematic creation and modification of artifacts, as well as manual transformations among models.

In addition to describing the information, it is also essential to make it accessible at an organizational level. Thus, using any artifact can help in this task, but this process depends on an organizational routine that systematically adopts the artifact’s use.

Our perception aligns with that of Whittle et al. (2014) and Hutchinson et al. (2014), which suggests that companies should consider the more holistic benefits that MDE can bring, rather than focusing solely on code generation. We understand that companies need to improve their use of models and model transformations, especially in the final stages of development.

Finally, an important finding is that the use of a model-based approach has a positive impact on organizations. Using these artifacts and technologies helps minimize some common issues that occur in knowledge-intensive organizations. For instance, the use of automatic or semi-automatic transformations aids in code maintenance, code standardization, productivity, and code understanding. In addition, the systematic use of models and transformations positively impacts reduced dependence on key employees, making it easier to find solutions to recurring issues and ensuring traceability.

7 Threats to Validity

We discuss the possible validity concerns based on Kitchenham et al. (2015) and Wohlin et al. (2012) in terms of construct, internal, external, and conclusion validity concerns, as well as the steps we have taken to minimize or mitigate them.

Construct validity: Encompasses design and generalization capacity of the study (Kitchenham et al., 2015) and is concerned with the extent to which the objects of study truly represent the theory behind the survey (Wohlin et al., 2012). To avoid mono-operation bias, we collected data from different sources (different companies with distinct sizes and focuses). Despite all companies operating in Brazil, many of them are actually based in another country, and many are not Brazilian companies.

Internal validity: Ability to describe the stages to conduct the study (Kitchenham et al., 2015) and reflects whether all causal relations are studied or if unknown factors affect the result (Wohlin et al., 2012). We defined a survey process based on the literature to ensure the study follows a reliable method. The survey instrument was validated through internal validation, pilot testing, and a pre-test to ensure that the instrument presents a correct sequence, covers all the necessary questions to answer the research questions, and that the questions are understandable to the respondents. Despite the rigorous instrument design and validation process, many participants did not finish answering the instrument. We consider that two factors are responsible for the rate of answers: first, due to the instrument's complexity. However, the complexity was necessary since we sought to gain a deeper understanding of how companies utilize model approaches and how these models relate to knowledge management (KM) theories. Second, many participants who did not complete are developers. Hence, the respondent should understand all software development processes to identify artifacts used in each phase, and many participants may be unable to do so. Thus, although the study presents this limitation, we consider it a side effect, necessitating the development of an instrument that enables the extraction of sufficient data to achieve the research objective.

External validity: External validity is concerned with the extent to which the results of this study can be generalized (Wohlin et al., 2012). It addresses factors related to research causal relationships (evolution and replication) (Kitchenham et al., 2015). To mitigate the impact of potential dominant participant numbers in a specific sector, the survey has been distributed to software professionals nationwide via email and professional social networks across various industrial sectors. Therefore, we have made every effort to reach subjects with diverse backgrounds, representing the software industry. Furthermore, we used a non-probabilistic sampling design; thus, external validity is limited. To address this, we reported demographic information of the participants and companies covered in our study. We cannot guarantee the reliability of all responses, as it is necessary to assume that the respondent accurately represents the company's reality. This fact may be considered a threat in any survey.

Conclusion validity: The ability to describe correct conclusions concerning the study performed (Kitchenham et al., 2015). We believe our results and findings are valid, supported by a rigorous process and careful data analysis. We attempted to reduce the bias for each research question by seeking support from the statistical results. Moreover, we developed analysis themes, and based on the survey results, we analyzed these themes to answer the research questions. Additionally, we compared our results with other literature sur-

veys, which provided greater confidence in the findings and supported our analysis.

8 Conclusion and Future Research

This study aimed to investigate the impact of MBE on knowledge management. To this end, a survey was conducted among Brazilian software development companies. The findings suggest that the use of artifacts and techniques in the MBE helps capture and organize information.

One of the goals of this work was to provide researchers with a roadmap of artifact models used by companies, called the MBE4COK roadmap. The roadmap, which graphically depicted the analyzed panorama from the use of artifacts in software development companies, provided readers with the ability to identify which requirement techniques, artifacts, and transformation types were adopted to promote the capture and organization of information in the surveyed companies.

Additionally, by utilizing the roadmap and analyzing the survey results, we addressed the research questions. Different artifacts are practical in various knowledge management concepts, considering the distinct phases of software development. For the requirements phase, user stories, use case models, and design phases, class diagrams, flowcharts, activity diagrams, and sequence diagrams are the most helpful artifacts regarding knowledge management (KM).

In software development, it is common to employ practices from agile methods, such as XP and Scrum, to facilitate knowledge management Kavitha and Irfan Ahmed (2011) and Santos et al. (2011). Furthermore, in conjunction with these practices, the use of requirement-gathering techniques, such as brainstorming and interviews, during the requirement phase can also help promote a better understanding of information, a crucial aspect of knowledge management.

Some companies reported that they do not often use artifact models, as they believe it is time-consuming to update such models when changes occur during the project. Some companies also believe agile methodologies, such as Scrum, are enough for knowledge management. However, as reported by previous studies (Ouriques et al., 2018; Khalil and Khalil, 2020), in companies using agile methods, knowledge management emphasizes informal communication, resulting in a significant amount of knowledge lost or not properly transferred to other individuals, and, instead of propagating the knowledge, it remains inside a few individuals' minds. Thus, based on the results of this study, we consider that using artifacts in conjunction with agile practices is an effective strategy for knowledge management, as many KM aspects can be facilitated through models and transformations.

Although the companies surveyed benefit from using artifacts, they do not fully utilize their potential. This vision is shared by Whittle et al. (2014), who consider that companies could benefit from a more holistic use of models. Thus, this survey showed that companies do not fully explore the benefits of manual, semi-automatic, or automatic transformations. The use of transformations facilitates the understanding and sharing of knowledge. Moreover, transformations enhance traceability, a crucial factor for knowledge manage-

ment (KM) in the software development process.

The use of models also helps to minimize common problems in knowledge-intensive organizations. Additionally, it offers benefits such as code standardization and increased productivity, while also facilitating code maintenance and comprehension.

This study has shown that MBE can enhance the capture and organization of knowledge. To better understand how the artifacts, techniques, and transformations discussed here aid KM, we plan to conduct a more in-depth descriptive study based on personal interviews and observations. It is also recommended to establish a framework to facilitate the adoption of the correct artifacts, techniques, tools, and transformations in each development phase. It utilizes the minimum number of models possible to prevent model-based development from becoming a burden, thereby aiding in knowledge management and traceability.

The sustainability of models throughout the software life-cycle remains a significant challenge, particularly during maintenance phases. Many companies struggle to keep models up to date and aligned with evolving code, which limits their long-term usefulness. Future research should focus on developing methods, processes, and tool support that enable effective synchronization between models and code, reducing manual effort and ensuring consistency. Additionally, while MBE has demonstrated value in managing explicit knowledge, its potential for capturing tacit knowledge, such as design rationale, architectural decisions, and domain expertise, remains underexplored. Understanding how models can represent this implicit knowledge could strengthen their role as long-term organizational memory.

Furthermore, broader empirical studies are needed to verify whether the findings of this research apply in different regions, industries, or organizational contexts. Comparative studies could help identify cultural and organizational factors that influence the successful adoption of MBE for knowledge management. Another critical research direction is exploring how KM practices and MBE can be better integrated into agile and hybrid development approaches, ensuring knowledge continuity in iterative environments.

Acknowledgements

This study was partly financed by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, Brazil), grant # 9908272120340724. Edson Oliveira Jr thanks CNPq/Brazil Grant #311503/2022-5.

References

- ABES, S. (2024). *Brazilian Software Market: Scenario and Trends*. Associação Brasileira das Empresas de Software.
- Agner, L. T. W., Soares, I. W., Stadzisz, P. C., and Simão, J. M. (2013). A brazilian survey on uml and model-driven practices for embedded software development. *Journal of Systems and Software*, 86(4):997–1005.
- Akdur, D., Garousi, V., and Demirörs, O. (2018). A survey on modeling and model-driven engineering practices in the embedded software industry. *Journal of Systems Architecture*, 91:62–82.
- Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araújo, J., Biffl, S., Cabot, J., Cortellessa, V., Fernández, D. M., Moreira, A., et al. (2019). Dealing with non-functional requirements in model-driven development: A survey. *IEEE Transactions on Software Engineering*, 47(4):818–835.
- American Productivity and Quality Center (2025). Levels of knowledge management maturity. <https://www.apqc.org/>. Accessed: May 2025.
- Bardin, L. (1977). *L'analyse de contenu*. Presses universitaires de France.
- Bennett, J., Cooper, K., and Dai, L. (2010). Aspect-oriented model-driven skeleton code generation: A graph-based transformation approach. *Science of Computer Programming*, 75(8):689–725.
- Bjørnson, F. O. and Dingsøyr, T. (2008). Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Inf. Softw. Technol.*, 50(11):1055–1068.
- Brambilla, M., Cabot, J., and Wimmer, M. (2017). *Model-Driven Software Engineering in Practice*. Synthesis Lectures on Software Engineering. Morgan Claypool Publishers.
- Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101.
- Braun, V., Clarke, V., and Rance, N. (2014). *How to use thematic analysis with interview data*, pages 183–197.
- Chen, P. P.-S. (1977). The entity-relationship model: a basis for the enterprise view of data. In *Proceedings of the June 13-16, 1977, national computer conference*, pages 77–84.
- Dalkir, K. (2005). *Knowledge Management in Theory and Practice*. Elsevier Butterworth-Heinemann, Boston, MA.
- Darimont, R., Delor, E., Massonet, P., and van Lamsweerde, A. (1997). Grail/kaos: an environment for goal-driven requirements engineering. In *Proceedings of the 19th international conference on Software engineering*, pages 612–613.
- Davenport, T. and Prusak, L. (1998). *Working Knowledge: How Organizations Manage What they Know*. Harvard Business School.
- 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 International Conference 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.
- Flora, H. (2014). A systematic study on agile software development methodologies and practices. *International Journal of Computer Science and Information Technologies*, 5:3626–3637.
- 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*, page 27–32. Association for Computing Machinery.
- France, R. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, page 37–54, USA. IEEE Computer Society.
- Garvin, D. A. (1988). *Managing Quality: The Strategic and Competitive Edge*. Free Press.
- Ghazi, A. N., Petersen, K., Reddy, S. S. V. R., and Nekkanti, H. (2019). Survey research in software engineering: Problems and mitigation strategies. *IEEE Access*, 7:24703–24718.
- Gorschek, T., Tempero, E., and Angelis, L. (2014). On the use of software design models in software development practice: An empirical investigation. *J. Syst. Softw.*, 95:176–193.
- Greenacre, M. J. (2010). Correspondence analysis. *WIREs Computational Statistics*, 2(5):613–619.
- Grossman, M., Aronson, J. E., and McCarthy, R. V. (2005). Does uml make the grade? insights from the software development community. *Information and Software Technology*, 47(6):383–397.
- Haynes, S. N., Richard, D. C. S., and Kubany, E. S. (1995). Content validity in psychological assessment: A functional approach to concepts and methods. *Psychological Assessment*, 7:238–247.
- Hutchinson, J., Whittle, J., and Rouncefield, M. (2014). Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, 89:144–161.
- Hutchinson, J., Whittle, J., Rouncefield, M., and Kristoffersen, S. (2011). Empirical assessment of mde in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, page 471–480. Association for Computing Machinery.
- Iandoli, L. and Zollo, G. (2008). *Organizational cognition and learning*. Hershey: Information Science Pub.
- Jolak, R., Savary-Leblanc, M., Dalibor, M., Vincur, J., Hebig, R., Pallec, X. L., Chaudron, M., Gérard, S., Polasek, I., and Wortmann, A. (2022). The influence of software design representation on the design communication of teams with diverse personalities. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*, pages 255–265.
- Jolak, R., Savary-Leblanc, M., Dalibor, M., Wortmann, A., Hebig, R., Vincur, J., Polasek, I., Le Pallec, X., Gérard, S., and Chaudron, M. R. (2020). Software engineering whippers: The effect of textual vs. graphical software design descriptions on software design communication. *Empirical software engineering*, 25:4427–4471.
- Josse, J., Husson, F., and Pages, J. (2010). Principal component methods - hierarchical clustering - partitional clustering: why would we need to choose for visualizing data? pages 1–17.
- Kavitha, R. and Irfan Ahmed, M. (2011). A knowledge management framework for agile software development teams. In *2011 International Conference on Process Automation, Control and Computing*, pages 1–5.
- Khalil, C. and Khalil, S. (2020). Exploring knowledge management in agile software development organizations. *International Entrepreneurship and Management Journal*, 16(2):555–569.
- Kitchenham, B. A., Budgen, D., and Brereton, P. (2015). *Evidence-Based Software Engineering and Systematic Reviews*. Chapman amp; Hall/CRC.
- Kleppe, A. G., Warmer, J. B., and Bast, W. (2003). *MDA explained: the model driven architecture: practice and promise*. Addison-Wesley Professional.
- Kobayashi, O., Kawabata, M., Sakai, M., and Parkinson, E. (2006). Analysis of the interaction between practices for introducing xp effectively. In *Proceedings of the 28th International Conference on Software Engineering*, page 544–550. Association for Computing Machinery.
- Kolovos, D. S., Paige, R. F., and Polack, F. A. C. (2006). On-demand merging of traceability links with models. In *ECMDA 06 Traceability Workshop*, pages 1–9.
- Kotonya, G. and Sommerville, I. (1996). Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18.
- KPMG Consulting (2000). Knowledge management research report 2000: Knowledge management maturity model (kmmm). Internal report, widely cited but not publicly available.
- Kulkarni, U. R., Ravindran, S., and Freeze, R. (2006). A knowledge management success model: Theoretical development and empirical validation. *Journal of management information systems*, 23(3):309–347.
- 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.
- Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Guest editors’ introduction: Model-driven development. *IEEE Software*, 20(05):14–18.
- Menolli, A., Cunha, M. A., Reinehr, S., and Malucelli, A. (2015). “old” theories, “new” technologies: Understanding knowledge sharing and learning in brazilian software development companies. *Information and Software Technology*, 58:289–303.
- Menolli, A., Reinehr, S., and Malucelli, A. (2013). Organizational learning applied to software engineering: a systematic review. *International Journal of Software Engineering and Knowledge Engineering*, 23(08):1153–1175.
- Molléri, J. S., Petersen, K., and Mendes, E. (2020). An empirically evaluated checklist for surveys in software engineering. *Information and Software Technology*, 119:106240.
- Moreira, A., Mussbacher, G., Araújo, J., and Sánchez, P. (2022). Theme section on model-driven requirements engineering. *Software and Systems Modeling*, 21(6):2109–2112.
- Nevis, E. C., Dibella, A. J., and Gould, J. M. (1995). Understanding organizations as learning systems. *Sloan Management Review*, 36:342–367.
- Nonaka, I., Toyama, R., and Konno, N. (2000). Seci, ba and leadership: a unified model of dynamic knowledge creation. *Long Range Planning*, 33(1):5–34.

- Nugroho, A. and Chaudron, M. R. (2008). A survey into the rigor of uml use and its perceived impact on quality and productivity. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, page 90–99. Association for Computing Machinery.
- OMG (2003). Mda guide version 1.0.1. online. <http://www.omg.org/cgi-bin/doc?omg/03-06-01>.
- Ouriques, R., Wnuk, K., Gorschek, T., and Berntsson Svensson, R. (2018). Knowledge management strategies and processes in agile software development: A systematic literature review. *International Journal of Software Engineering And Knowledge Engineering*, 29(3):345–380.
- Ras, E., Memmel, M., and Weibelzahl, S. (2005). Integration of e-learning and knowledge management – barriers, solutions and future issues. In *Proceedings of the Third Biennial Conference on Professional Knowledge Management*, page 155–164, Berlin, Heidelberg. Springer-Verlag.
- Rodríguez-Sánchez, J.-L., González-Torres, T., Montero-Navarro, A., and Gallego-Losada, R. (2020). Investing time and resources for work–life balance: The effect on talent retention. *International journal of environmental research and public health*, 17(6):1920.
- Rus, I. and Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software*, 19(3):26–38.
- Santos, V., Shinoda, A. C. M., Goldman, A., and Fishcher, A. (2011). A view towards Organizational Learning: An empirical study on Scrum implementation. *23rd International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, pages 583–589.
- Schmidt, D. (2006). Guest editor’s introduction: Model-driven engineering. *Computer*, 39(2):25–31.
- Senge, S., Kleiner, A., Roberts, C., Ross, R., and Smith, B. J. (1994). *The fifth discipline field book*. Doubleday.
- Sommerville, I. (2011). *Software Engineering*. Addison-Wesley Publishing Company.
- Stahl, T. and Voelter, M. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley Sons Ltd.
- Störrle, H. (2017). How are conceptual models used in industrial software development? a descriptive survey. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, EASE’17*, page 160–169, New York, NY, USA. Association for Computing Machinery.
- Tiwana, A. (2002). *The knowledge management toolkit*. Prentice Hall PTR.
- Tomassetti, F., Torchiano, M., Tiso, A., Ricca, F., and Reggio, G. (2012). Maturity of software modelling and model driven engineering: A survey in the italian industry. In *16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012)*, pages 91–100. IET.
- Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., and Reggio, G. (2011). Preliminary findings from a survey on the md state of the practice. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 372–375.
- Torchiano, M., Tomassetti, F., Ricca, F., Tiso, A., and Reggio, G. (2013). Relevance, benefits, and problems of software modelling and model driven techniques—a survey in the italian industry. *Journal of Systems and Software*, 86(8):2110–2126.
- Vasanthapriyan, S., Tian, J., and Xiang, J. (2015). A survey on knowledge management in software engineering. In *2015 IEEE International Conference on Software Quality, Reliability and Security-Companion*, pages 237–244. IEEE.
- Wang, X. and Cheng, Z. (2020). Cross-sectional studies: strengths, weaknesses, and recommendations. *Chest*, 158(1):S65–S71.
- Whittle, J., Hutchinson, J., and Rouncefield, M. (2014). The state of practice in model-driven engineering. *IEEE Software*, 31(3):79–85.
- Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., and Heldal, R. (2017). A taxonomy of tool-related issues affecting the adoption of model-driven engineering. *Software & Systems Modeling*, 16:313–331.
- Wohlin, C. and Aurum, A. (2015). Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Softw. Engg.*, 20(6):1427–1455.
- Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.
- Yanzer Cabral, A. R., Ribeiro, M. B., and Noll, R. P. (2014). Knowledge management in agile software projects: A systematic review. *Journal of Information & Knowledge Management*, 13(01).