**INTELI - INSTITUTO DE TECNOLOGIA E LIDERANÇA**

DAYLLAN DE SOUZA ALHO

GIOVANNA FURLAN TORRES

VITÓRIA RODRIGUES DE OLIVEIRA

# FELIX
# Framework for Structuring Solutions and Integrating Experiences

SÃO PAULO

2025

DAYLLAN DE SOUZA ALHO

GIOVANNA FURLAN TORRES

VITÓRIA RODRIGUES DE OLIVEIRA

# FELIX

# Framework for Structuring Solutions and Integrating Experiences

Final Paper presented to the Instituto de Tecnologia e Liderança (INTELI) as a prerequisite for obtaining a bachelor's degree in Information Systems.

Supervisor: Vanessa Nunes

Prof.ª Vanessa Nunes (Supervisor)

Dayllan de Souza Alho (Student)

Giovanna Furlan Torres (Student)

Vitória Rodrigues de Oliveira (Student)

SÃO PAULO

2025

# ACKNOWLEDGMENTS

The completion of this work was made possible thanks to the support of many people and institutions, to whom we express our deepest gratitude.

To our advisors and professors, we are grateful for their dedication, patience, and for sharing their knowledge and experience with us, guiding us throughout this journey. To our friends and family, we thank you for the encouragement, support during challenging moments, and for believing in us.

This work is a reflection of collective effort and the positive impact that education can generate when it receives the investment and trust of those who believe in the future.

## ABSTRACT

The goal of this project is to develop FELIX – Framework for Structuring Solutions and Integrating Experiences, a methodology designed for Product Owners (POs) to enhance their autonomy in defining the solution architecture. Currently, this responsibility falls exclusively on the Tech Leader, which can create communication bottlenecks and impact transparency, deadlines, and scope. FELIX aims to provide guidelines enabling POs to map systems, anticipate constraints, and align solutions with business needs and technical requirements.

**Keywords:** Product Owner, Solution Architecture, Tech Leader, Framework, Product Management, Agile Methodology.

## LIST OF ACRONYMS

PO: Product Owner

## SUMMARY

# 1. Introduction

During their internship at “company x”, the authors of this paper identified a recurring challenge in the development of systems within the agile methodology: during the refinement of demands, the definition of the solution's architecture was concentrated on the Tech Lead, without the active participation of the Product Owner (PO), which made it difficult to understand the product and transition the tasks to the developers. Although POs are responsible for prioritizing the backlog and aligning deliveries with business needs, the lack of a clear direction in the organization of the architecture results in difficulties in the predictability of the process, in communication with stakeholders and in making strategic decisions before technical implementation.

This gap leads to problems such as misalignment between business expectations and technical constraints, rework due to unstructured architectural decisions and delays in delivery times. Often, Tech Leaders end up taking responsibility for the entire architecture, which overloads these professionals and limits the autonomy of the POs. This scenario compromises the efficiency of the teams, affecting deadlines, scope and transparency in communication.

With this in mind, this paper proposes the development of a framework that allows POs to map and organize the solution's architecture in a more strategic and autonomous way. With clear guidelines, this framework will make it easier to anticipate technical restrictions, better understand system integrations and reduce the need for constant reviews with Tech Leaders. First step, described in this document, involves interviews with POs and Tech Leaders, as well as a benchmarking study to analyze practices adopted by other companies.

## 1.1. Problem

At “company x”, the absence of a structured model for Product Owners to actively participate in defining the solution's architecture creates a series of challenges. The lack of transparency in defining new projects and the revision of previously established agreements due to technical issues can impact deadlines, scope and alignments with stakeholders. In addition, the centralization of architectural decisions in Tech Leaders can result in bottlenecks and reduce the agility of the development process.

This problem highlights the need for a framework that enables POs to define the solution's architecture efficiently and strategically, ensuring greater predictability and risk mitigation.

## 1.2. Objectives

The aim of this work is to develop a framework that allows product owners to structure the solution's architecture autonomously, reducing exclusive dependence on technical leaders and ensuring greater predictability, transparency and alignment between the business vision and architecture decisions.

### 1.2.1. Specific Objectives

- Identify the challenges faced by POs in managing solution architecture through bibliographic research and benchmarking;

- Develop guidelines and tools to help POs define architectural solutions aligned with business needs;

- Create a structured model for documenting and communicating the solution architecture among stakeholders;

- Test and validate the framework in real scenarios, assessing its impact on project management and the predictability of value delivery;

- Refine and consolidate the framework based on the feedback obtained, ensuring its applicability in different organizational contexts.

# 2. Methodology

This study adopts a qualitative and exploratory approach, based on action research, to investigate and develop a framework that expands the role of the Product Owner (PO) in solution architecture. The choice of action research is justified by its iterative and participatory nature, allowing challenges to be analyzed in successive cycles of planning, action, observation and reflection (LEWIN, 1946).

The methodology is structured in three main stages:

1. Literature review - analysis of academic studies and market practices on the role of the PO in architectural decisions.
2. Collection of empirical data - interviews with POs and Tech Leaders to map challenges and opportunities.

3. Developing and validating the framework - applying the principles of action research to continually adjust the proposed framework.

Below we detail the action research adopted and its practical application in this study.

## 2.1 - Action Research Methodology

Action research was first proposed by Kurt Lewin (1946) and is characterized by a continuous cycle of observation, planning, action and reflection. This method allows for active intervention in the problem studied, enabling the empirical validation of solutions and their adaptation as new discoveries emerge throughout the process.

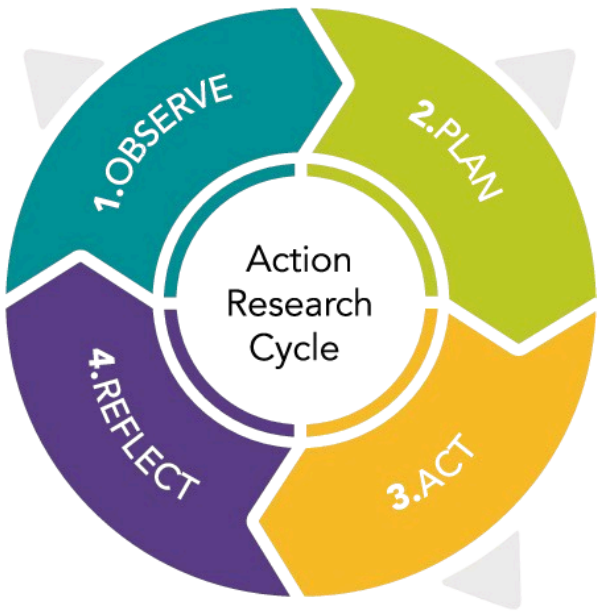
According to McNiff (2017) and Coghlan & Shani (2018), action research drives organizational change by integrating researchers and participants in the investigative process, ensuring that the solutions developed are contextually relevant and applicable. In addition, studies such as those by Brydon-Miller et al. (2019) and Greenwood (2022) highlight the importance of the active participation of stakeholders, reinforcing the need for the solutions developed to be co-created to ensure their long-term sustainability.

The methodological cycle adopted in this study follows the four classic stages of action research:

1. Observation - identifying emerging patterns and assessing the impact of the proposed strategies.
2. Planning - theoretical survey and identification of the challenges faced by POs in solution architecture.
3. Action - interviews and analysis of practices adopted by organizations with autonomy models for POs.
4. Reflection and Adjustments - analysis of the data collected and continuous refinement of the framework.

Figure 1 illustrates this methodological cycle applied to the research:

Figure 1 – The Action Research Cycle and stages



*Source: Researchgate, López Franco, 2020*

This iterative approach ensures that the framework is developed empirically and validated in practice, promoting greater transparency in decision-making and alignment between business needs and architectural definitions.

## 2.2 Action Research Approach

With an iterative research approach, the study is moving into the practical phase, involving professionals from the sector to validate the challenges and needs identified. The research is being conducted in a progressive and adaptive manner, allowing for continuous adjustments as new data is collected.

The empirical research included interviews with market professionals, such as Product Owners and Tech Leaders, with the aim of understanding the challenges and needs related to the definition of daily processes and responsibilities in the architecture. Based on the data obtained, the research follows a structured path for the initial development of the theoretical framework. The aim is to ensure that the framework is aligned with the real demands of the market, promoting greater transparency in decision-making and reducing risks in systems development.

The main methodological aspects applied to the study include:

1. Identification of real problems faced by POs in their interaction with the solutions architecture.
2. Collection and analysis of empirical data through interviews and review of organizational processes.
3. Iterative refinement of the framework, ensuring alignment with market practices.

This process makes it possible to continually validate the proposed model, ensuring that it is theoretically grounded, empirically validated and applicable to the reality of the agile environment.

As highlighted by McNiff (2017) and Coghlan & Shani (2018), action research drives organizational change by integrating researchers and participants into the process, ensuring that the solutions developed are practically relevant and applicable to real-world scenarios. In this study, action research is used to test and refine the framework in collaboration with POs and agile technical leaders, making it more adaptable to industry needs.

One of the key challenges in this approach is the potential researcher influence on results. To mitigate bias, structured forms and standardized data collection techniques were employed, allowing interviewees to express their experiences autonomously. This ensures that the findings reflect the real needs of practitioners rather than researcher assumptions.

According to Brydon-Miller et al. (2019) and Greenwood (2022), action research emphasizes active stakeholder participation in co-creating solutions. This methodology enables the study to not only understand existing challenges faced by POs but also to collaboratively build and validate a framework that addresses the gaps between business strategy and technical architecture.

The following figure presents an illustration of the action research cycle, highlighting the continuous iteration across planning, action, observation, and reflection. By integrating literature review, empirical insights, and action research cycles, this methodology ensures that the developed framework is theoretically sound, empirically validated, and practically applicable to the agile context.

# 3. Project Context

This project is being developed based on the internal processes of “company X”, seeking to optimize collaboration between the different functions within the agile environment. The clear definition of roles and responsibilities, in line with the agile methodologies adopted by the company, is fundamental to guaranteeing the efficiency of the process and the success of deliveries.

## 3.1. Specific Roles

The roles within an agile environment are fundamental to the success of a project, as they guarantee the effective application of the framework's principles and practices. Each role is responsible for specific tasks which, when performed well, contribute to optimizing the team's performance and delivering consistent value to the client. Clearly defining these roles is important to creating a collaborative, high-performance structure that is aligned with the project's strategic objectives. The roles described below are based on the processes of “company X,” aligning with its operational needs and internal practices.

### 3.1.1 Role of the Scrum Master

The Scrum Master acts as a facilitator and coach for the team, ensuring the correct application of Scrum practices. According to Schwaber and Sutherland (2020), “the Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide, helping everyone to understand the theory, practices, rules and values of the framework” (p. 6). Their role is essential for the formation of a cohesive and productive team, ensuring the adoption of agile principles.

In addition, Scrum Masters face significant challenges in organizational environments that are not yet fully adapted to the agile mindset. Hoda and Noble (2017) point out that “Scrum Masters often face difficulties in balancing their facilitator role with the leadership expectations imposed by more traditional organizations” (p. 45). This balance is important for the Scrum Master to perform their role effectively, without compromising the autonomy and collaboration of the team.

Studies indicate that the Scrum Master's experience can directly influence team performance. Moe et al. (2019) point out that “the presence of an experienced Scrum Master can significantly increase team cohesion, resulting in more consistent deliveries aligned with customer objectives” (p. 12). This positive impact reinforces the importance of the Scrum Master in facilitating communication and removing impediments that could compromise the team's productivity.

At company X, there is no specific person dedicated to the role of Scrum Master. Instead, this role is mainly performed by the Tech Lead, who takes responsibility for facilitating the application of Scrum practices, guiding the team and ensuring that the agile principles are followed. The Tech Lead, as well as leading the technical part and getting directly involved with the stakeholders, also plays the role of facilitator, removing impediments and promoting a collaborative environment, which ensures that the team runs smoothly and that deliveries are aligned with the project's objectives.

### 3.1.2 Role of the Product Owner

The Product Owner, in turn, is responsible for maximizing the value of the product developed by the team. According to Cohn (2019), “the Product Owner must ensure that the backlog is always up to date, prioritized and aligned with the organization's strategic objectives, serving as the main bridge between stakeholders and the development team” (p. 23). This position requires advanced communication, negotiation and prioritization skills to balance different interests and ensure high-value deliveries.

However, playing this role is not without its challenges. Pichler (2018) states that “one of the biggest challenges facing Product Owners is balancing the conflicting demands of stakeholders while maintaining a focus on the value of the product for the end customer” (p. 34). As such, the ability to make strategic decisions without compromising the product vision is essential to the success of the project.

Effective communication between the Product Owner and the development team is a determining factor in the successful execution of Scrum. Gregory et al. (2021) note that “clear communication between the Product Owner and the development team is essential to avoid misunderstandings and ensure that deliveries meet stakeholder expectations" (p. 18). This continuous alignment strengthens collaboration and contributes to an efficient workflow.

At company X, the Product Owner function is structured independently, with a dedicated team that is not directly integrated with the developer team. Each Product Owner is assigned to manage one or more teams, and is responsible for coordinating and ensuring that demands are carried out effectively. The role of this function is fundamental as a bridge between the business team and the developers, facilitating communication and the flow of information. The Product Owner ensures that priorities are clearly defined, activities are constantly updated and business objectives are aligned with the product's progress and deliveries. This allows the company to meet market needs in an agile and strategic way, without losing focus on the value delivered to the end customer.

### 3.1.3 Role of the Tech Lead

According to Unisinos (2024), the Tech Lead is the professional responsible for leading the technical part of a project, defining the technologies and architectures used. They also act as a link between the development team and other departments in the company, promoting alignment and ensuring that compatible methodologies are adopted. Their role goes beyond carrying out technical tasks, taking a guiding role in identifying and solving problems.

Alura (2024) stresses that the Tech Lead must have a balance between technical and interpersonal skills. Their main duties include defining the technical roadmap, supervising the team to ensure consistent deliveries and creating a collaborative environment. In addition, they must align technical goals with the company's strategic objectives, acting as a bridge between technical teams and stakeholders.

To perform this role effectively, according to Unisinos (2024), it is essential that the Tech Lead has advanced knowledge of programming languages such as Python and Java, experience with agile methodologies (Scrum or Kanban) and interpersonal skills such as clear communication and effective leadership. These factors are important to the success of projects and strategic alignment within organizations.

At company X, the Tech Lead's role is even more comprehensive and strategic. They are responsible for multiple areas and play a collaborative role, going beyond technical leadership to actively engage with stakeholders. In partnership with the Product Owner (PO), the Tech Lead is responsible for building a roadmap that is coherent and viable for the development team, aligning the needs of the business with the capabilities of the team. In addition, the Tech Lead manages various project fronts, ensuring that initiatives are conducted in an integrated and efficient manner, promoting continuous product progress with a clear and aligned vision between the technical and business teams.

### 3.1.4 Role of the Stakeholder

Stakeholders are individuals, groups or organizations with an interest in the project. They can be internal (such as staff and managers) or external (such as clients and suppliers). According to Strong (2022), understanding and managing their expectations is important to aligning objectives, minimizing risks and maximizing benefits.

They can be classified as internal, including employees and managers, or external, encompassing customers, investors and communities. Primary stakeholders are directly impacted, such as end users, while secondary stakeholders have an indirect interest, such as regulators. Euax (2022) points out that identifying key stakeholders is essential for prioritizing efforts, as they have the greatest influence on the project.

Effective stakeholder management involves identifying, analyzing and engaging them. According to Docusign, tools such as stakeholder mapping make it easier to define roles, responsibilities and levels of influence, helping to avoid conflicts and strengthen relationships.

At company X, the stakeholders are mainly responsible for understanding product evolution needs and ensuring that changes are aligned with the strategic objectives of the business. This role is played by business representatives, who monitor the market and customer demands. They are responsible for passing on the necessary developments to the Tech Lead and Product Owner (PO), who in turn translate this information into technical development and the definition of product priorities. In this way, the stakeholders act as key agents in defining the product's direction and aligning the market's needs with the team's capabilities.

### 3.1.5 Role of the Solutions Architect

The solution architect plays an essential role in agile projects, being responsible for translating business needs into effective technical solutions. They act as a link between stakeholders and development teams, ensuring that the system's architecture meets both functional and non-functional requirements, such as performance, security and scalability. According to Gertel (2021), this professional must have a strategic vision to define architectural guidelines, as well as promoting an incremental and evolutionary design that allows adaptations according to the project's needs.

In addition, the solutions architect plays an important role in stakeholder engagement, ensuring that all interested parties share a clear vision of the project's objectives. According to Menon, Sinha and MacDonell (2021), this continuous interaction between the architect and stakeholders prevents misalignments and allows for more assertive technical decision-making. Another relevant aspect is their technical leadership, since they act as mentors to the development team, sharing knowledge, promoting good practices and helping to solve complex problems. As Altmann (2021) points out, this mentoring contributes to the team's professional growth and improves the quality of deliveries.

Collaborative decision-making is also part of their responsibilities, as the solution architect must encourage the active participation of the development team, ensuring that technical choices are feasible and aligned with the team's capacity. This also involves the selection of suitable technologies, often involving the creation of Proofs of Concept (PoCs) to validate the viability of a solution before its full implementation (Arquitetocloud, 2021). In addition, quality assurance and risk management are fundamental functions of this professional, who needs to anticipate possible flaws in the architecture and implement mitigation strategies to ensure the stability and security of the solution.

At company X, there is no specific position dedicated to the solutions architect. Instead, this role is shared between the Tech Lead and the business team. The Tech Lead, with the strategic support of the business team, plays a crucial role in defining the system architecture, and is responsible for ensuring that the technical solutions meet the demands of the project, while respecting the technical limitations and capabilities of the team.

## 3.2. POs' Main Responsibilities

The theoretical foundation of this study is based on a literature review on the responsibilities and challenges of the Product Owner (PO) in agile environments, with an emphasis on Scrum. While there are various perspectives on the role of the PO, this study specifically focuses on the challenges highlighted by Schwaber (2004), Kristinsdóttir (2014), and Sverrisdottir, Ingason, and Jonasson (2014). These references support the analysis of how the role of the PO has evolved in different organizational contexts and how it can face the challenges inherent in its function. It is clear that throughout the existence of this study, new references may emerge and contribute to further discussions and confront different approaches over time.

There are multiple perspectives on the role of the Product Owner in agile methodologies, and different frameworks and organizations interpret this role in various ways. However, this study focuses on the challenges faced by POs in real-world applications, as reported in the selected references. By narrowing the scope to these specific challenges, this research aims to provide a deeper understanding of the practical difficulties POs encounter, rather than attempting to cover all possible variations of the role.

The role of the Product Owner (PO) has emerged as one of the most important within agile methodologies, being responsible for ensuring that product development is aligned with business objectives and user needs. According to Schwaber (2004), the PO represents the interests of the stakeholders and defines the prioritization of the product backlog, guaranteeing the delivery of continuous value. This role is multifaceted and involves both interaction with stakeholders and direct collaboration with development teams.

Sverrisdottir, Ingason, and Jonasson (2014, p. 258) point out that, in practice, the interpretation of the PO's role varies considerably between organizations. Some companies adopt a model in which there are two Product Owners for a single product, with one responsible for the business aspects and the other for the technical aspects of development. This approach can facilitate communication between stakeholders and technical teams but also deviates from the original Scrum model, which defines the PO as a single person responsible for all these areas. The Product Owner is, among many things, responsible for establishing and communicating a clear vision of the product to the entire team. As described by Kristinsdóttir (2014), this responsibility includes both communicating a strategic vision and constantly adapting to changes in the market and the organization.

### 3.2.1. Product Backlog Management

The PO is responsible for prioritizing the product backlog, ensuring that the most valuable features are delivered first. Schwaber (2004) points out that this prioritization is based on both customer needs and return on investment (ROI). At Spotify, the POs reported difficulties in directly measuring the value of the work done, since many deliveries were to internal customers. Some Product Owners used qualitative feedback and indirect metrics, such as stakeholder satisfaction and user base growth, to measure the effectiveness of the features delivered (Kristinsdóttir, 2014, p. 9).

Sverrisdottir, Ingason and Jonasson (2014, p. 263) observed that the frequency of backlog review and maintenance varies significantly between the POs interviewed. In some cases, the backlog is reviewed daily, while in other organizations this activity occurs only once a month:

> "Review of the PB is very different between the participants. Most people seem to look at the backlog weekly, daily or even many times a day, as there may be significant changes on the backlog. For one participant this is done only once per month, when a product is put into operation" (Sverrisdottir, Ingason and Jonasson, 2014, p. 263).

### 3.2.2 Engagement with Stakeholders

The Product Owner acts as a bridge between the stakeholders and the development team. According to Kristinsdóttir (2014, p. 7), "the Product Owners did agree that the teams should know who their customer is, but they did not all find it to be their responsibility to identify those customers and their needs or to communicate those needs to their team." This shows that there is a variation in the perception of the PO's responsibilities within different organizations.

Sverrisdottir, Ingason and Jonasson (2014, p. 264) reinforce that managing expectations is one of the main duties of the PO, and that their communication skills are fundamental to maintaining alignment between the technical teams and the demands of the business:

> "Most participants indicated that expectations management was one of the most important duties of the PO. Examples of vital characteristics of the PO to be able to fulfill their role are planning and communication skills. This was considered necessary to create a common understanding between the business manager and the IT department." Sverrisdottir, Ingason and Jonasson (2014, p. 263)

### 3.2.3 Facilitating Collaboration with the Development Team

Agility depends on effective communication between the PO and the development team. Kristinsdóttir (2014, p. 10) points out that one of the challenges faced by Product Owners at Spotify was the limited time for interaction with their teams. Some POs were able to devote only 5% of their time to direct collaboration, while others spent up to 70% of their time working side by side with developers. This variation had a direct impact on the clarity of demands and the speed of deliveries.

Sverrisdottir, Ingason and Jonasson (2014) complement this analysis by pointing out that the time dedicated by POs to interactions with the team varies considerably. While some POs spend most of their time aligning priorities with stakeholders, others work more closely with developers, ensuring that the technical specifications are clear and aligned with the product vision (Sverrisdottir, Ingason and Jonasson, 2014, p. 262).

### 3.2.4 Decision Making and Prioritization

The Product Owner (PO) must make quick, data-driven decisions to ensure the effectiveness of agile development. However, as noted by Kristinsdóttir (2014, p. 9), there are significant challenges in this practice, as many POs reported difficulties in accessing objective metrics and quantitative data, leading them to rely on intuition and qualitative feedback. "Most of the Product Owners found it difficult to measure the value of the work their team was doing because often there was no transaction of money taking place" (KRISTINSDÓTTIR, 2014, p. 9).

Spotify tracks usage and user satisfaction as impact metrics for changes to the software. However, as many of these changes are incremental, users don't always notice the changes made, which makes it difficult for POs to assess the real impact of the teams' work (Kristinsdóttir, 2014, p. 9-10).

Sverrisdottir, Ingason and Jonasson (2014, p. 263) point out that the autonomy of the PO in making decisions can be a determining factor in the success of the project:

> "All participants said that it is either fairly important or very important that the PO has unrestricted authority to make decisions in the project. The role of product owner is to make decisions and this is often essential to be able to finish the project." Sverrisdottir, Ingason and Jonasson (2014, p. 263).

### 3.2.5 Managing expectations and communication

An important aspect of the PO's role is managing expectations and aligning visions between the different stakeholders. As highlighted by Kristinsdóttir (2014, p. 13), "Product Owners need to have one foot in the daily work and one foot in the future, and to lead people to work on the right things at any given moment." This reinforces that the role requires a constant balance between operational execution and the strategic vision of the product. Sverrisdottir, Ingason and Jonasson (2014, p. 265) add that the PO needs to be a communicative and influential leader, as their ability to negotiate priorities and align expectations between stakeholders and technical teams is a critical factor in the success of the project.

## 3.3. Main Challenges

The Product Owner is responsible for establishing and communicating a clear vision of the product to the entire team, ensuring that the backlog is properly prioritized and that deliveries are aligned with the organization's strategic objectives. However, the literature shows that the practical application of this role varies significantly between organizations, which creates challenges for its effective execution.

In the BTG context, there is evidence of challenges related to communication between POs and developers, as well as the participation of POs in the solution architecture, which does not seem to be a consolidated organizational practice. This issue still needs to be investigated further to understand its real extent and impact. The literature shows that similar challenges occur in different organizations. Kristinsdóttir (2014, p. 10) observes that the limitation of the time dedicated by POs to interacting with development teams can directly impact the clarity of demands and the speed of deliveries: “Every Product Owner found it extremely important to spend time with the team every day, as much as they physically could, but some struggled with that as they worked with more than one team and had to attend various meetings.”

Another recurring challenge pointed out in the literature is the lack of a standardized understanding of the role of the PO within the organization. Sverrisdottir, Ingason and Jonasson (2014, p. 257) point out that this variability in the interpretation of the PO's role is common in different companies, leading to difficulties in the efficient implementation of Scrum and in backlog decision making.

The difficulty in measuring the value generated by deliveries is also an obstacle present in many organizations. Kristinsdóttir (2014, p. 8) points out that, in the absence of direct monetary transactions, many POs find it difficult to objectively measure the impact of deliveries. This problem can manifest itself when there are no uniformly established success metrics between the different teams, making it difficult to effectively prioritize the backlog.

The management and review of the backlog also represent relevant challenges. Sverrisdottir, Ingason and Jonasson (2014, p. 263) observed that “review of the PB is very different between the participants. Most people seem to look at the backlog weekly, daily or even many times a day, as there may be significant changes on the backlog. For one participant this is done only once per month, when a product is put into operation.”

Another critical factor addressed in the literature is the autonomy of the PO in decision-making. Sverrisdottir, Ingason and Jonasson (2014, p. 264) point out that “all participants said that it is either fairly important or very important that the PO has unrestricted authority to make decisions in the project. The role of product owner is to make decisions and this is often essential to be able to finish the project.” The lack of this autonomy can lead to delays and compromise the ability to adapt agilely to the market.

Finally, managing expectations and efficient communication between stakeholders and the development team are key challenges. Sverrisdottir, Ingason and Jonasson (2014, p. 264) point out that “most participants indicated that expectations management was one of the most important duties of the PO. Examples of vital characteristics of the PO to be able to fulfill their role are planning and communication skills.” Kristinsdóttir (2014, p. 13) reinforces the importance of continuous communication by stating that “Product Owners need to have one foot in the daily work and one foot in the future, and to lead people to work on the right things at any given moment”.

Thus, it can be seen that the challenges faced by POs in different organizational contexts involve aspects such as communication, backlog prioritization, autonomy and participation in the solution architecture. In the case of BTG, investigating these challenges will allow us to better understand the specificities of this environment and how these factors can impact the efficiency of the PO's role in product development.

# 4. Benchmarking

Market benchmarking is a strategy that allows organizations to compare their performance with that of competitors or industry leaders, in order to identify gaps and improve processes. According to Oliveira and Silva (2018), “benchmarking is an essential tool for companies to learn from the best practices in the market, adapting them to their context to improve their competitiveness”.

## 4.1 Expert Interviews

The aim of this section is to present the research carried out. The main focus is to understand the daily work of the project's actors, how demands are organized and refined, as well as to analyze their responsibilities and the degree of autonomy they have.

A preliminary survey was carried out during sprint 2 (17/02 to 28/02) with the developers and POs. During sprint 3 (03/03 to 14/03), the questions applied to these actors were reviewed in order to identify possible gaps or points for improvement. In addition, the final version of the questionnaire for Tech Leads and Solution Architects was drawn up. In sprint 4 (17/03 to 28/03), the aim is to hold an official interview with the three groups of actors and apply the final questionnaire, covering POs, developers, and Tech Leads / Solution Architects.

### 4.1.1 Interview with developers

To ensure that the development process is efficient and that demands are met clearly and precisely, it is important to involve developers in researching and mapping their activities. They are often the ones responsible for carrying out the tasks, so understanding how demands reach them and how they are interpreted is essential. A good understanding of the needs and challenges faced by developers can optimize workflow, avoid misunderstandings and promote more efficient communication between teams. By identifying possible flaws in the way demands are passed on, it is possible to adopt improvements that have a positive impact on productivity and the quality of product development.

The aim of this research was to understand how work demands are assigned in the day-to-day lives of professionals, focusing in particular on the flow of tasks for developers. It sought to understand the channels through which demands reach them, such as through platforms like Monday, Jira or Trello, or through direct communication with leaders, Tech Leads, Product Owners (PO), or other means. In addition, the study sought to assess whether the current process is effective or could be improved, and how any changes could impact product development.

The questions submitted to the participants were:

1. How do work demands come to you on a daily basis? Describe the process, including where you receive it and how this assignment takes place.

2. Do you think the way demands come to you currently works well or could it be improved?
    a. It works well, I don't see any need for change.
    b. It works, but it could be better.
    c. It doesn't work well, it needs improvement.

3. If you think it could be improved, how could this process be made more efficient?

4. How often do you receive work demands?
    a. Daily
    b. Weekly
    c. Monthly
    d. Other…

5. Do you feel you have the autonomy to organize your own demands or are you totally dependent on others?
    a. I have complete autonomy
    b. I have some autonomy, but I follow a fixed direction
    c. I am totally dependent on others

The answers to both surveys are available in Appendix C. The following is a comparative analysis of these responses, with the aim of identifying patterns, divergences and relevant perceptions about the process of receiving demands from developers.

Each section of the analysis corresponds to a specific question in the survey - for example, Analysis 1 “Origin and flow of demands” refers to Question 1, and so on - ensuring a structured, question-oriented comparison.

**Analysis question 1 - Origin and flow of demands**

Most of the demands come through direct communication channels, such as face-to-face or virtual meetings, emails and management tools such as Jira, Azure DevOps and Confluence. The responses indicate a strong dependence on communication tools such as Teams and the practice of daily meetings to align priorities. However, there is a lack of a centralized or unified system for monitoring all demands, which can lead to confusion and a lack of transparency between teams. Tasks are often assigned in a fragmented way, either by teams of Tech Leads or Product Owners (POs), or even on the team members' own initiative.

Although some teams use task management tools, such as Jira and Azure DevOps, many responses indicate that there is still a gap in the structured and transparent process of assigning demands. Some teams receive demands directly from leaders, without a clear formalization of priorities or deadlines, which is aggravated by communication often taking place via informal meetings and the use of dispersed tools. This suggests the absence of a formal model for managing delivery expectations and ensuring that everyone involved has visibility of the progress of tasks.

On the other hand, some answers indicate that employees show autonomy and proactivity, taking the initiative to look for new tasks, either by asking leaders directly or by analyzing the task boards themselves. This shows that, although there is a hierarchical structure, many teams have a culture of proactivity. However, a lack of clarity regarding the prioritization of these tasks can result in activities that are not aligned with the most urgent needs of the business.

Another relevant point is the lack of visibility and transparency in the workflow between teams. The lack of a centralized system and continuous communication makes visibility difficult, especially in larger projects involving multiple teams or stakeholders. Although some project management tools are used, many teams don't seem to have a consolidated view of the progress of tasks throughout the organization, which makes it difficult to monitor the progress of demands.

| Standard | Explanation | Impact |
|---|---|---|
| **Communication through direct channels** | Demands usually come in through meetings (face-to-face or virtual), emails or chat tools. | It can lead to a disorganized task assignment process without a centralized vision. |

| | | |
|---|---|---|
| **Use of management tools** | Some teams use tools like Jira, Azure DevOps and Confluence to organize their work. | It helps to structure tasks, but can be insufficient when there is no constant follow-up. |
| **Lack of formalization in the process** | Tasks are often assigned informally, with little documentation or clear prioritization. | It can lead to a lack of transparency and visibility in the progress of the work, undermining management. |
| **Dependence on Tech Leads and POs** | Most tasks are assigned by Tech Leads or POs, limiting the autonomy of the teams. | It reduces agility, since developers depend on an extra layer of approval. |
| **Proactivity in assignments** | Some developers take the initiative to seek out tasks or offer solutions directly. | It can improve individual productivity, but it can cause misalignment with the overall strategy. |

## Analysis question 2 - Efficiency of the reception process

The responses reveal a clear trend regarding the perception of the current effectiveness of the process for receiving work demands. The majority of participants (13 out of 15) believe that the process works, but could be better. However, some responses indicate that the current process does not work well and needs improvement. In addition, there is a small group (3 people) who believe that the current way of receiving demands works well and there is no need for change.

The majority of respondents believe that the process is viable, but that there is room for improvement. This suggests that, despite some flaws or areas that could be optimized, the current model is not completely ineffective. However, the need for improvement may be linked to issues such as lack of visibility, formalization of attributions or an overload of unstructured communication. There are also responses indicating that the process is not fully meeting expectations, possibly due to the lack of a centralized system or communication failures between the parties involved (stakeholders, technical leaders and teams). This may be leading to a more disorganized workflow, compromising efficiency. On the other hand, the low number of responses that consider the current process to be effective suggests that a more structured model could be more effective. However, some people believe that the process is already sufficient for the size and complexity of the teams or projects. These answers probably come from teams with a smaller scope or less complex demands.

| Standard | Explication | Impact |
|---|---|---|
| **"It works well, I don't see any need for change."** | The majority of responses point out that although the process works, there are points for improvement. | It signals the need for adjustments to increase efficiency and improve communication, without radical changes. |
| **"It works, but it could be better."** | A smaller number of participants consider that the current process meets the team's needs well. | It indicates a well-established process that is working for the size and complexity of the team. |
| **"It doesn't work well, it needs improvement."** | Some responses indicate that the task assignment system is not working efficiently and needs significant adjustments. | It reflects the need for more in-depth reviews of the demand management process, possibly involving better communication tools or practices. |
| **Concern about micromanagement** | One participant's response indicates that excessive control can be detrimental, leading to a decrease in team morale. | It warns of the risk of over-management, which can have a negative impact on team performance and motivation. |

## Analysis question 3 - Process improvements

Most respondents recognize that the process of receiving and assigning demands could be improved. Below are some recurring suggestions that could be implemented to make the process more efficient:

The use of systems that automatically organize demands after meetings was a frequent suggestion. Assigning tasks using more effective tools, such as integration with systems like Jira, Azure DevOps or others, was also mentioned. In addition, automating the sending and follow-up of tasks is a suggestion that came up to improve the process. These suggestions indicate the need to centralize the process in a more efficient task management platform, which allows for better integration between all the stages, from raising the demand to execution. Tools that can automatically organize and prioritize tasks would bring more agility and visibility to the process, reducing the dependence on manual communication.

Another recurring suggestion was to improve communication and collaboration. More direct interaction with stakeholders would help to better understand demands and clarify doubts. In addition, the active participation of developers in the discovery process could increase visibility of business needs and the impact of changes. More detailed specifications on the flows and impacts of changes were also pointed out as an important improvement. Improving communication between developers and stakeholders would help ensure that demands are aligned with real needs and reduce the risk of rework. In addition, allowing developers to participate more actively in the initial phases would help increase their understanding of the requirements and scope, which would generate a more efficient process.

With regard to the organization and definition of tasks, it was suggested that the activities be organized in such a way that they reflect the skills and experience of the developers, as well as creating more detailed and specific tasks, with a clear step-by-step approach, to make it easier for those involved. It was also mentioned that creating cards with all the information needed to carry out the tasks, avoiding the need to search for additional data, would be beneficial. Organizing tasks according to the experience and skills of the developers allows for more efficient allocation and improves workflow. More detailed and well-described tasks reduce the time spent searching for information and allow developers to start working more efficiently, reducing delivery time.

Improving visibility and transparency was also highlighted. It was suggested to increase the visibility of tasks and deadlines by creating clear systems for notifying and following up on activities. Centralizing tasks on a platform where developers can signal when they have started or completed an activity was also mentioned. In addition, monitoring the progress of tasks in real time was highlighted as an important improvement. Improving visibility and transparency in tasks helps avoid confusion and makes it easier to track progress, ensuring that everyone involved can see the progress of activities, quickly identify blockages and optimize communication between teams.

Finally, adjustments to the planning process were suggested. Improving the definition of the scope of changes and assessing the impact of each change was pointed out, as was the creation of a committee to assess risks and plan broader changes that impact several projects or teams. Having a clearer and more detailed planning process, with risk assessment, would help to improve accuracy in defining deadlines and scope. This type of process would also reduce the risk of work overload and help to adjust expectations.

| Standard | Explication | Impact |
|---|---|---|
| **Centralization and Automation** | Integrated task management tools that automatically organize and prioritize demands. | Increased visibility, organization and efficiency in the process of assigning and monitoring tasks. |
| **Direct communication with stakeholders** | Developers involved in the discovery process and in direct contact with stakeholders to better understand demands. | Reduced risk of rework, increased understanding of requirements, and alignment of expectations. |
| **Organizing and Detailing Tasks** | More detailed tasks, with clear steps aligned with the developers' skills. | Reduced time spent searching for information, increased efficiency and clarity in the work carried out. |
| **Visibility and Transparency in Tasks** | Creating centralized platforms with clear notifications about the status of tasks and deadlines. | Better monitoring of progress, less risk of tasks not being completed on time. |
| **Clear prioritization of tasks** | To-do list with clearly prioritized tasks to ensure that the most important ones are done first. | Increased productivity and focus on tasks with greater impact and urgency. |
| **Planning and Risk Assessment** | Better definition of the scope of changes, assessment of impacts and creation of a committee for risk analysis. | More efficient planning, with less risk of surprises or work overload for the teams. |

## Analysis question 4 - Frequency of demands

Analysis of the responses reveals that the majority of respondents receive daily demands, which is a reflection of the dynamic nature of the work environment, where new challenges arise frequently, especially in areas of development and project management. Some people mentioned other frequencies, and this gives us an idea of how different teams or functions can have different rhythms.

The vast majority (8 out of 15 people) receive tasks on a daily basis, which is to be expected in agile development environments with a high volume of work. This may indicate that the teams have a constant workload and need to be continually updated on priorities and tasks. For these teams, the fluidity of the task assignment process is crucial to maintaining the pace of production without interruption.

Some respondents mentioned that they receive demands on a weekly basis. This suggests that the process of planning or reviewing tasks occurs with this regularity, which may be related to weekly sprints or alignment meetings. This frequency indicates that the workload is more predictable, with a defined schedule of deliveries and tasks to be carried out throughout the week.

Only one person mentioned that they receive demands on a monthly basis. This may be indicative of less dynamic work or of an area with longer production cycles, such as large-scale projects or those involving more strategic and long-term planning.

**Analysis question 5 - Autonomy in organizing demands**

The analysis of the answers reveals a diversity of levels of autonomy in the process of organizing work demands, which suggests different degrees of control that team members have over how tasks are assigned and carried out. The distribution of responses can be divided into the following categories:

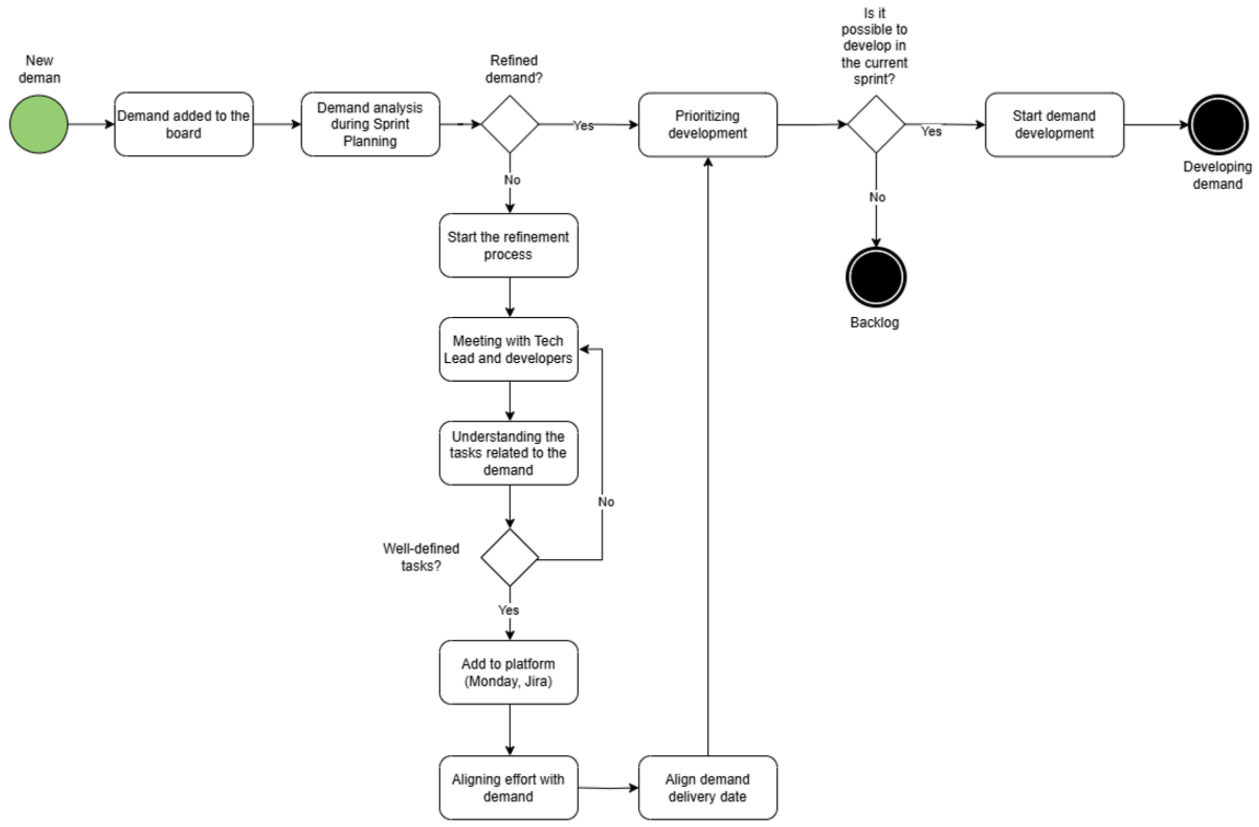
Seven people (approximately 47%) indicate that they have complete autonomy to organize their own demands. These people probably have a high level of responsibility, with the freedom to manage time and priorities on their own. This level of autonomy is positive, as it allows for greater flexibility and can increase motivation and engagement, allowing professionals to choose the tasks that best align with their skills and interests.

Seven people (approximately 47%) indicate that they have some autonomy, but still follow a fixed direction. These people probably have more control over their daily tasks, but still depend on some level of supervision or guidance, either from a manager or a more rigid structure for assigning tasks. This may indicate a balance between independence and the need for alignment with organizational goals and team coordination.

Only one person (approximately 7%) reported that they are totally dependent on the assignment of others to organize their demands. This suggests that this person is in a role where decisions about what to do, when to do it and how to do it are determined exclusively by others, possibly in a more hierarchical structure or with more centralized work processes.

**Macro flowchart based on research**

This flow describes the typical steps involved in receiving demands, based on a survey carried out with developers.


New deman
Demand added to the board
Demand analysis during Sprint Planning
Refined demand?
Yes
No
Start the refinement process
Meeting with Tech Lead and developers
Understanding the tasks related to the demand
Well-defined tasks?
No
Yes
Add to platform (Monday, Jira)
Aligning effort with demand
Align demand delivery date
Prioritizing development
Is it possible to develop in the current sprint?
Yes
No
Backlog
Start demand development
Developing demand

The flowchart constructed from the interviews represents, in macro form, the main aspects identified during the process of receiving and developing demands. Among the highlights observed is the way in which demands are initially added to the board (such as Jira or Monday) and analyzed in the sprint planning meetings. This initial stage reflects the reality of many teams that deal with demands coming from different channels, often in a decentralized way. Next, a check is made to see if the demand has already been refined, and if not, a refinement process begins.

This refinement usually involves a meeting with the Tech Lead and the developers, who take a more active stance, typical of the roles traditionally assigned to the Product Owner. The developers help to understand the details of the demand and the tasks involved. After this alignment, another check is made to ensure that the tasks are well defined. If they are, the demand is entered into the management platforms and the necessary effort is aligned, taking into account the available resources and complexity. This point in the process highlights the direct involvement of developers in both the planning and organization of deliveries, something that ideally should be more structured and centred on the PO.

Finally, after the development has been prioritized, it is checked whether it is possible to carry out the demand in the current sprint. If so, development begins; if not, the demand is sent to the backlog. This part of the process demonstrates the teams' attempt to maintain agility and a continuous pace of delivery, but it also reveals challenges related to the prioritization and visibility of activities.

## Assumption of PO Responsibilities by Developers

Analysis of the answers to the form revealed that developers often take on responsibilities that would ideally be assigned to Product Owners (POs). This overlap is largely due to the absence of well-defined processes, fragmented communication and a lack of detail in demands.

Among the main responsibilities of the POs that end up being absorbed by the developers is the effort to understand the demands. Many professionals report the need to seek out the context of the tasks on their own, analyzing boards or contacting stakeholders to clarify doubts, due to the lack of complete information at the time of transfer.

In addition, in some cases developers end up defining which tasks to carry out based on their own perception of priority, without formal guidance. It is also common for them to identify technical dependencies and align with other teams, functions that would normally fall to the PO during the planning phase.

Finally, developers often update the status of demands in the tools and monitor the progress of deliveries, activities that should be shared with the PO to ensure visibility and alignment with business objectives.

### 4.1.2 Interview with POs

The purpose of this section is to present the research carried out with the POs and their respective analyses. The study focuses on investigating the process of refining demands in the daily lives of POs, an essential stage in product management. This refinement consists of the detailed definition of demands, ensuring that the development teams have clear and precise information to implement the desired solutions. The research seeks to understand the origin of the demands, the agents involved in the process, the degree of autonomy of the POs in the refinement, as well as the methodologies and tools used for this purpose.

Two interviews were carried out with different POs, guaranteeing the anonymity of the participants. The first questionnaire was administered between 02/17 and 02/28, obtaining 5 responses, while the second took place between 03/17 and 03/28, with a total of 6 responses. The questions asked of the participants were as follows:

1. How do requests reach you? Describe where requests come from, who participates in this process and how demands are assigned.

2. When a demand arrives, is it already fully defined or do you need to refine it? Explain whether demands arrive ready-made or whether there is room for improvement.

3. How does the refinement of demands occur in your daily routine? Describe your refinement process, what steps you follow, who participates and what tools or methods you use.

4. Do you feel you have enough autonomy to refine demands in the best way? Why? Explain whether you have the freedom to adjust the demands as necessary or whether there are restrictions that hinder this process.

5. What are the main challenges you face when refining demands? List the main obstacles that hinder more efficient refinement, such as lack of information, time constraints, lack of alignment with stakeholders, etc.

The answers to the two surveys are available in Appendix A. The following is a comparative analysis of these responses, with the aim of identifying patterns, divergences and relevant insights into the process of refining demands in the day-to-day running.

Each section of the analysis corresponds to a specific survey question—for instance, Analysis 1 "Origin and flow of demands" refers to Question 1, and so on—ensuring a structured and question-oriented comparison.

**Analysis question 1 - Origin and flow of demands**

Demands can come from a variety of sources, including internal stakeholders (leadership, business areas), external stakeholders (customers) and internal teams (development, UX, operations). Some arrive in a structured way, through management tools and strategic planning, while others are passed on informally, via emails, WhatsApp or direct conversations.

The Product Owners' reports vary. Some describe a structured flow, with annual planning and periodic reviews, while others point to a disorganized scenario, with demands coming in chaotically. In some cases, prioritization takes place formally before Program Increments (PI), while in others, demands are received without clear prioritization criteria.

Market changes, competitive movements and emergency requests from customers often require replanning. In many responses, Tech Leads are primarily responsible for assigning demands and defining technical priorities. This scenario highlights one of the central challenges of the survey: the strong dependence on Tech Leads which can create bottlenecks and compromise the autonomy of POs, making predictability and transparency difficult in the decision-making process.

| Standard | Explanation | Impact |
|---|---|---|
| Demands come from multiple channels | E-mails, messages, meetings, agile tools, top-down leadership | Difficult to trace and standardize the flow |
| Lack of prior alignment on priorities | Some POs mention quarterly/annual planning, but others receive demands without clear criteria | Generates abrupt changes in scope and constant replanning |
| Stakeholders strongly influence prioritization | Demands come from commercial areas, customers, UX, operations and leadership | Can compromise the strategic vision if there are no well-defined criteria |
| Tech Leads act as gatekeepers | Tech Leads often assign demands and define effort | Reduces PO autonomy and can cause delivery bottlenecks |

**Analysis question 2 - Level of demand refinement**

Most respondents point out that demands rarely arrive fully refined and ready for development. In most cases, only the central theme is known, requiring detailed work to define business rules, technical impacts and actual scope.

Simple demands usually come better defined, while more strategic and complex demands require multiple meetings, alignments and iterations before being implemented. The time dedicated to refinement grows in proportion to the complexity and technical dependencies involved.

In addition, many demands require continuous interaction with clients, business areas, UX, development and other teams. The lack of a standardized format for delivering requirements contributes to rework in the refinement phase, making the process less efficient.

| Standard | Explanation | Impact |
|---|---|---|
| Demands come in without sufficient detail | Most demands come in with only a general theme, requiring more in-depth analysis | Generates rework and the need for additional meetings |
| Complexity impacts the level of refinement required | The more complex the demand, the more time and effort is spent on refinement | Delays in defining scope and technical impacts |
| Stakeholders play an essential role in refinement | Refinement requires frequent contact with technical, business and customer teams | It requires a structured alignment process to avoid misalignment |
| Lack of a standard for receiving requests | Some companies have frameworks such as AGNS, but most work with different formats | Hinders predictability and efficiency of refinement |

## Analysis question 3 - Refinement process

Most respondents point out that refinement involves multiple stakeholders and takes place iteratively over time. Simple demands can be refined directly by the PO, with little or no external interaction, while more complex demands require multiple meetings, detailed documentation and the participation of several teams.

To support refinement, tools such as Jira, Confluence, Monday, Figma and BPMN are used to document and manage the process. In addition, some answers mentioned the use of chat groups and recorded meetings to facilitate the exchange of information. Refinement usually starts with a business analysis and then goes through a technical evaluation. The participation of tech leads and specialists is often cited as essential for mapping impacts on systems and business rules.

At the end of the process, many teams formalize the decisions and definitions in structured documentation, such as Wikis, Confluence, flowcharts or cards in Jira. This record ensures alignment between those involved and reduces the risk of misinterpretations during development.

| Standard | Explanation | Impact |
|---|---|---|
| Refinement takes place in stages and involves multiple stakeholders | First business understanding → technical analysis → validation and formalization | Ensures clarity and reduces the risk of rework |
| Simple vs. complex demands require different approaches | Simple: quick refinement by the PO. Complex: multiple meetings and detailed documentation | Helps balance time invested in refinement |
| Tools structure and document refinement | Jira, Confluence, Monday, BPMN, Figma, flowcharts, emails and chats are all common. | Facilitates traceability and communication between areas |
| Refinement is a continuous process, not a single step | Occurs throughout the sprint/planning and can be adjusted as new insights emerge | Maintains alignment and avoids surprises in delivery |

## Analysis question 4 - Level of autonomy

Many POs report that their autonomy is limited, especially in demands involving the backend, integrations or architectural decisions. In these cases, dependence on Tech Leads is significant, as technical feasibility needs to be assessed before moving forward. In the case of more superficial demands, such as interface improvements or adjustments to internal processes, autonomy tends to be greater.

When the client is already clear about what they want or there are regulatory requirements, there is little room for adjustment. On the other hand, when the demand comes in unstructured, the PO has more freedom to refine it, as long as they can back up their decisions with solid arguments.

Some POs mention that the lack of adequate time for refinement negatively impacts the process, compromising the quality of strategic planning and the ability to better question stakeholder needs. Furthermore, even when they have the autonomy to refine demands, many feel the need to validate decisions with managers or Tech Leads before moving forward.

POs who feel more autonomous generally stress the importance of maintaining an open communication channel with clients and other areas. This continuous contact allows for adjustments throughout the process and reduces the risk of wrong decisions that need to be corrected later.

| Standard | Explanation | Impact |
|---|---|---|
| Autonomy depends on the technical complexity of the demand | The more technical the demand, the more dependence on tech leads and less freedom for adjustments | PO may feel lack of predictability and security in refinement |
| Regulatory or very specific demands have less flexibility | If a requirement comes ready-made from the client or is imposed by regulation, the PO has little room for refinement | PO acts more as a facilitator of the process than as a decision-maker |
| POs who document and argue well gain more autonomy | Explaining the reason for changes and documenting helps gain credibility with stakeholders | Facilitates changes in scope without major objections |
| POs who feel technically insecure tend to seek validation before moving forward | If a PO is unclear about the technical impact of a change, they need to consult tech leads or managers | This can lead to delays in refinement and increase the burden on tech leads |

## Analysis question 5 - Main challenges

POs face difficulties in obtaining technical details of requirements, either due to a lack of documentation or the constant need for interactions with IT teams. The absence of code documentation aligned with business documentation makes it difficult to verify requirements in an agile way and make informed decisions.

Business pressure for quick deliveries leads to incomplete refinements, which often continue during development. In addition, the limited availability of Tech Leads to support refinement results in less informed decisions and late adjustments.

Another recurring challenge is the arrival of demands without a well-defined purpose, prioritizing “what to do” without a clear understanding of “why”. This reduces the autonomy of the PO, who could contribute to defining the best solution if the objectives were more transparent. In addition, frequent changes of direction generate rework and loss of context for the team, since stakeholders are not always clear about what they really need.

A deeper understanding of the business, its internal politics, and strategic direction can help define better technical solutions—either directly, by aligning decisions with clear objectives, or indirectly, by anticipating constraints and stakeholder expectations.

POs also face difficulties in assessing the complexity of a demand before development begins. This lack of technical visibility compromises scope negotiation, making it difficult to analyze impacts and define viable alternatives. In addition, some changes can generate financial or contractual impacts that are not considered in the initial refinement, creating additional risks to the project.

**Convergences and divergences between the theory and practice of the PO's role**

The aim of this section is to revisit the points made about the responsibilities of Product Owners (POs) described in topic 3.2, based on the literature, and compare them with the data collected in the survey with POs. The intention is to identify convergences and divergences between theory and practice, understanding how the responsibilities attributed to OPs are manifested in the organizations analyzed.

- **Backlog management and prioritization:** The literature points out that the PO should have autonomy in prioritization based on business value and return on investment (Schwaber, 2004). However, in practice, many interviewees reported strong influence from stakeholders (leadership, commercial areas, operations), with prioritization often being defined outside of the PO's control. In some cases, the Tech Lead plays a determining role in the distribution of demands, which reduces the PO's autonomy - diverging from the model described by Schwaber and reinforcing the hybrid scenario described by Sverrisdottir et al. (2014), in which responsibilities are shared informally.

- **Involvement with stakeholders:** The theory emphasizes that the PO should be the link between the team and the stakeholders, communicating the needs of the users and the vision of the product (Kristinsdóttir, 2014). The research data shows that this function is present, but not always in a structured way. Some POs don't feel responsible for identifying stakeholders or translating their demands to the technical team, showing a gap between the ideal model and operational reality.

- **Collaboration with the development team:** Although Scrum proposes close working between the PO and the technical team, the study revealed a wide variation in the time dedicated by POs to direct collaboration with developers - reflecting the findings of Kristinsdóttir (2014). In environments where the PO actively participates in technical discussions, there is greater clarity in deliveries. However, in contexts where the PO has low availability or relies heavily on the Tech Lead, problems of alignment and incomplete refinement arise.

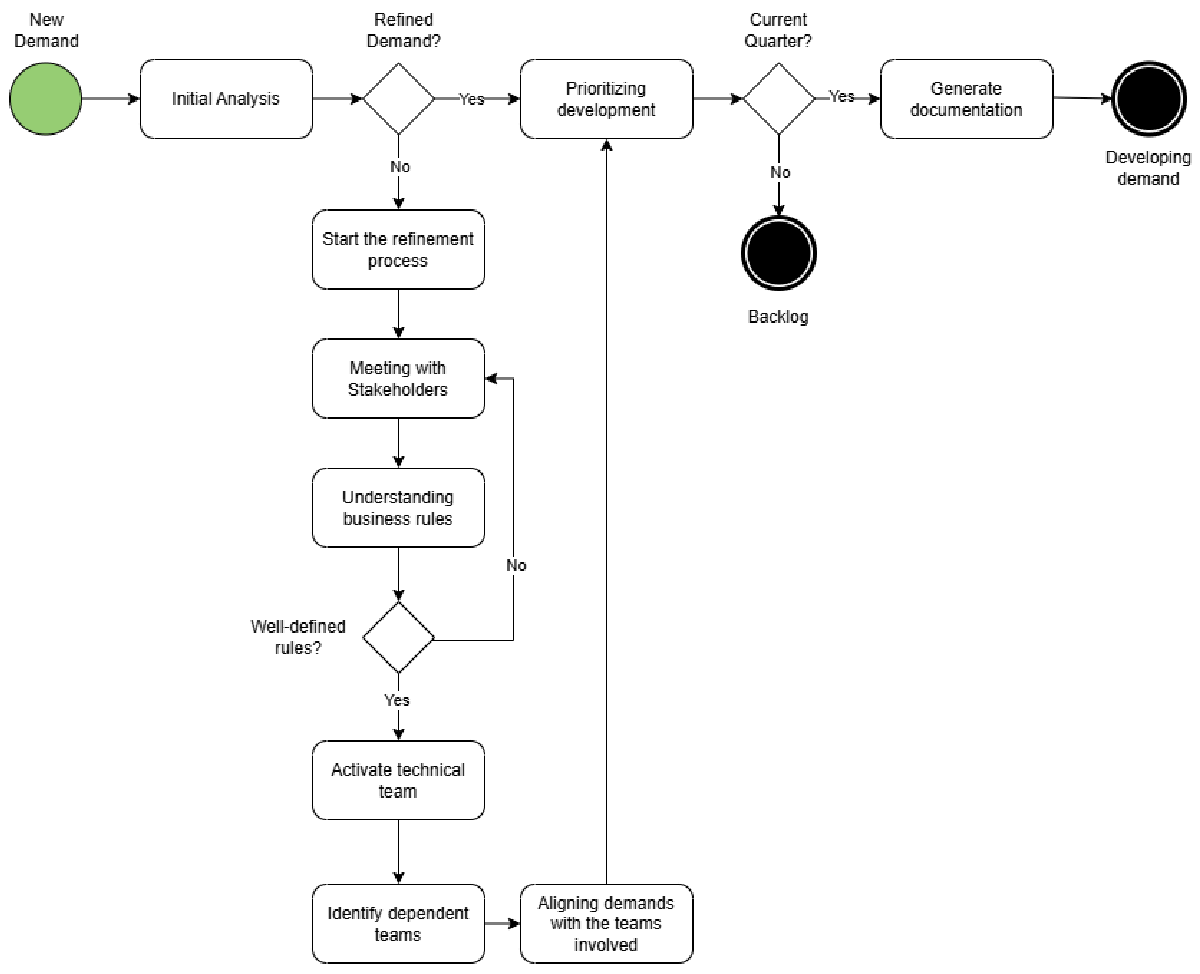
- **Autonomy in decision-making:** The literature reinforces the importance of the PO's autonomy for the effectiveness of the agile process. However, the reports show that this autonomy is often limited, especially when it comes to technical demands. The need for constant validation with Tech Leads and managers reflects an organizational structure that does not yet give the PO full decision-making power, which corroborates the challenges pointed out by Sverrisdottir et al. (2014) regarding the importance of decision-making authority for project success.

- **Communication challenges and expectations:** Managing expectations and the ability to negotiate are cited as essential skills in the literature. In practice, it was observed that POs face difficulties in this respect, mainly due to the pressure to deliver quickly and the lack of clarity in requests. Fragmented communication and the lack of clear criteria for prioritization reduce the effectiveness of the OP's role as a strategic articulator - a challenge already foreseen by Kristinsdóttir (2014).

- **Refinement process:** The iterative and collaborative refinement process described in the literature is partially confirmed in practice. Although the interviewees report the use of tools and methodologies to organize refinement, there is a lack of standardization and an excess of informality in the arrival of demands. This results in continuous rework and refinement during development, contrary to the idea of solid preparation prior to the start of implementation.

## Macro flowchart based on research

This flow describes the typical steps involved in refining demands, based on research carried out with Product Owners (POs).



Source: Authors

The flowchart constructed from the interviews represents, in macro form, the main aspects identified in the work of the Product Owners throughout the process of arriving at and developing demands. Among the highlights observed:

- **Detailed Refinement:** The POs dedicate part of their time to refining the demands, which involves exploring different scenarios, identifying dependencies and understanding the business rules.
- **Extensive Collaboration:** The diagram also shows the strong collaborative action of the POs with different stakeholders - such as business areas, technical leaders and other teams - reinforcing the role of continuous articulation throughout the demand cycle.
- **Expectations management:** The active management of expectations in relation to scope and deadlines, mentioned frequently in the interviews, is reflected in the interactions provided in the flowchart, showing the points of communication and alignment with stakeholders throughout the process.

### 4.1.3 Interview with tech Leads.

This section presents the analysis of the third round of interviews conducted between March 17 and March 28, focusing on the role of Solution Architects and Tech Leads in receiving, refining, and executing technical demands. The research seeks to understand how architectural decisions are made, how collaboration with Product Owners and stakeholders is structured, and what challenges arise in aligning business needs with scalable and maintainable technical solutions.

The six participants interviewed in this phase of the research shared their practices, constraints, and approaches to balancing fast-paced delivery with architectural integrity. Each subsection below corresponds to one of the survey questions, with observed patterns organized in the format: Standard, Explanation, and Impact.

Below are the questions that were aimed at this audience in question, the Teche Leaders

1. How do technical demands reach you? Describe the process for receiving technical demands and how these demands are communicated to you (meetings, management platforms, direct communication with stakeholders, POs, etc.).
2. How do you define and prioritize the technical approach for each demand you receive? What criteria do you use to decide on the best technical solution to

meet the demand? How do you decide between different approaches or technologies?

3. Do you have the autonomy to define the architecture and technical solutions to the demands or are there limitations? Are there external factors (e.g. budget constraints, deadlines, stakeholder requirements) that affect your ability to define the ideal solution?

4. What tools or methodologies do you use to coordinate the execution of technical demands with your team? Tell us about the tools such as Jira, Azure DevOps, Trello, among others, that you use to manage the technical execution of demands.

5. Have you ever faced challenges related to changing requirements or priorities during the development cycle? How do you deal with unexpected changes and how do they affect the proposed technical solution? How do you adjust the team's planning to adapt?

6. What is your collaboration process with POs and stakeholders throughout the development cycle? When do you interact with POs or stakeholders during the process? How do these interactions impact the definition and execution of technical solutions?

**Analysis question 1 – Origin and flow of technical demands**

Technical demands come from a variety of sources: product owners, internal stakeholders (leadership, business teams) and even from the architects or developers themselves when they identify technical or improvement opportunities. These demands come through structured tools such as Confluence or Azure DevOps, but also through informal means such as meetings and chats.

| Standard | Explanation | Impact |
|---|---|---|
| Technical demands come from multiple channels | Includes PO requests, business input, and direct identification by architects or devs | Requires constant triage and prioritization to avoid overload |
| Some demand sources bypass standard intake processes | Direct stakeholder requests or self-initiated improvements are common | Leads to context-switching and potential conflicts in prioritization |

| | | |
|---|---|---|
| Lack of standardization in how demands arrive | Use of diverse tools and informal communication | Reduces traceability and hinders alignment across teams |

**Analysis question 2 – Criteria for defining the technical approach**

Architects and Tech Leads describe a decision-making process that weighs reusability, complexity, scalability, and operational costs. Many rely on predefined templates or cloud-native patterns to ensure fast and reliable delivery—especially in tactical contexts. Evaluation criteria include business impact, time-to-market, and future maintainability.

| Standard | Explanation | Impact |
|---|---|---|
| Preference for reusable and standardized architectures | Use of known patterns (e.g., Lambdas, Cronjobs, managed services) | Speeds up delivery and reduces risk |
| Decisions driven by complexity vs. value vs. time | Balancing technical effort with business return | Helps justify trade-offs and define MVPs |
| Focus on long-term scalability and maintenance | Includes operational cost, system growth, and monitoring | Ensures sustainability of the solution |

**Analysis question 3 – Autonomy in defining the solution**

While Tech Leads generally have autonomy, organizational constraints (non-approved technologies, leadership bias, cost ceilings) influence decision-making. Even with architectural freedom, decisions are often guided by internal standards and delivery targets.

| Standard | Explanation | Impact |
|---|---|---|
| Autonomy varies depending on delivery context | Tactical projects allow more freedom than strategic, highly visible ones | Requires careful navigation of priorities and expectations |
| Internal politics and tech preferences constrain options | Certain tools are avoided due to leadership taboos or lack of approval | Limits experimentation and architectural evolution |
| Alignment with internal standards fosters consistency | Solutions must conform to team norms for easier collaboration | Promotes knowledge sharing and supportability |

**Analysis question 4 – Tools and coordination methods**

The use of platforms such as Azure DevOps, Confluence, Monday, and OneNote is widespread for coordinating execution. However, the lack of integration between business specs and technical documentation is a recurring issue, creating extra effort during refinement and onboarding.

| Standard | Explanation | Impact |
|---|---|---|
| Execution managed through multiple tools | Azure DevOps for tracking, Confluence for specs, OneNote for notes | Fragmented visibility of the full process |
| Separation between business and code documentation | Limited linkage between what's built and why it was built | Reduces context for new developers or auditors |
| Methodologies adapted to team needs | Varying levels of adherence to Scrum or Kanban | Influences team discipline and predictability |

**Analysis question 5 – Communication of technical priorities**

Translating technical implications into business language is a core challenge. Misunderstandings often arise around integration complexities and the effort involved in seemingly "simple" tasks. Tech Leads often take on the role of educators, guiding POs and stakeholders toward feasible timelines and expectations.

| Standard | Explanation | Impact |
|---|---|---|
| Tech complexities not fully understood by non-technical teams | Integration paths (e.g., API vs. messaging) affect timelines significantly | Causes misalignment and underestimated effort |
| Stakeholder involvement is uneven | Some engage in early refinement, others only during delivery phases | Inconsistent scoping and late feedback loops |
| Communication depends on individual initiative | Tools help, but personal follow-up remains essential | Increases cognitive load and time spent outside coding |

**Analysis question 6 – Handling requirement changes during development**

Change is constant—requirements evolve mid-sprint due to missed initial details or new business needs. Tech Leads react by replanning and negotiating scope. Some teams formalize scope changes with stakeholders; others use iterative delivery as a buffer.

| Standard | Explanation | Impact |
|---|---|---|

| | | |
|---|---|---|
| Frequent scope changes during development | Due to unclear initial specs or new business inputs | Leads to rework and delivery delays |
| Refinement continues during implementation | Teams adapt on the go, sometimes without full revalidation | Compromises solution integrity and planning |
| Formalization helps manage expectations | Changes are negotiated with stakeholders and scheduled in phases | Improves transparency and avoids conflict |

## Convergences and divergences between the theory and practice of the Tech Lead’s role

This section analyzes the alignment between the theoretical expectations of the Tech Lead’s role and the empirical findings from the interviews with professionals who perform this function in agile environments. The objective is to highlight convergences and divergences between theory and practice, focusing on the daily responsibilities, constraints, and decision-making challenges faced by Tech Leads in the context of software development.

### Architectural vision and technical leadership

According to Unisinos (2024), the Tech Lead is expected to define the technologies and architectures used in a project, promoting technical alignment and ensuring the sustainability of the system. In practice, interviewees confirmed this role, often acting as key figures in technical planning. However, they also reported that architectural decisions are sometimes compromised by tight deadlines or delivery pressures, leading to technical shortcuts that may increase future maintenance efforts.

### Autonomy and organizational constraints

Although Alura (2024) emphasizes that the Tech Lead should have autonomy to define technical paths aligned with strategic goals, the interviews revealed that this autonomy is often limited by internal politics, leadership preferences, or the non-approval of certain technologies. Even with experience and leadership, some Tech Leads must negotiate with other stakeholders to justify architectural choices, which transforms technical decisions into political negotiations.

### Collaboration with stakeholders and POs

The role of the Tech Lead also includes strong collaboration with stakeholders, a point reinforced by Menon, Sinha, and MacDonell (2021), who highlight the importance of continuous engagement to ensure alignment of expectations. The interviewees reported that this collaboration is often informal and highly dependent on personal initiative. They play a fundamental role in "translating" technical complexities into business language, helping Product Owners and stakeholders understand the real scope and implications of architectural decisions.

## Team coordination and technical mentoring

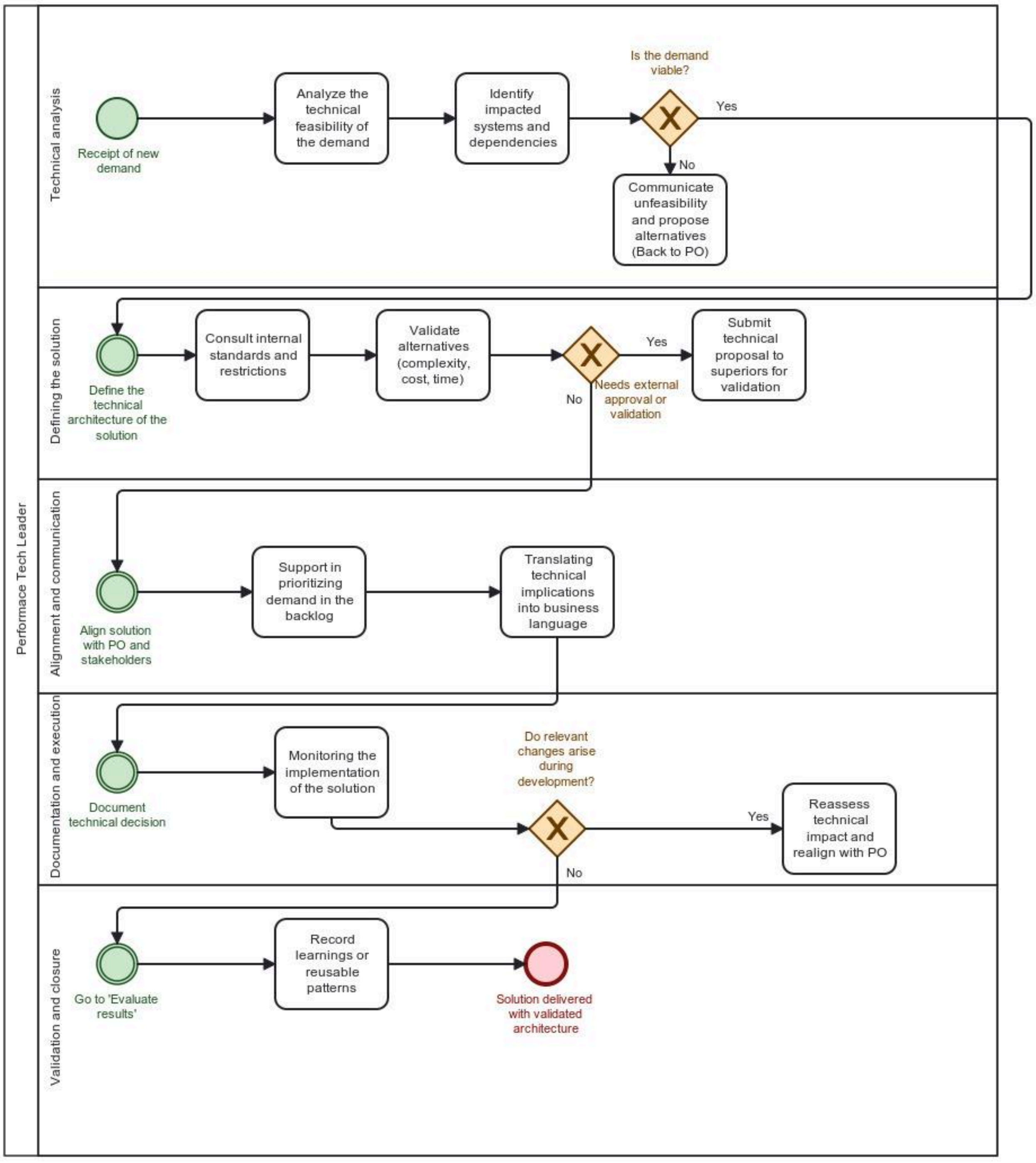
As stated by Altmann (2021), the Tech Lead acts as a mentor to the development team, encouraging knowledge sharing and best practices. This is confirmed in practice: many participants describe themselves as both team leaders and enablers of technical growth. However, some also mentioned difficulties in dedicating time to mentoring due to urgent deliveries or the accumulation of strategic responsibilities beyond their technical role.

### Adaptation to changing requirements

The literature points out, particularly in Arquitetocloud (2021), that the solution architect must be prepared to adapt architectures to changing business needs, often validating approaches through Proofs of Concept (PoCs). This flexibility is also observed in practice. Tech Leads reported frequent mid-sprint changes in requirements, which forces them to replan technical solutions and negotiate scope. While some teams formalize these changes, others adapt on the fly, which can compromise architectural integrity.

## Macro flowchart based on research

In addition to the qualitative analysis of the interviews, it was possible to structure a macro-process that represents the flow of the Tech Lead's work based on the responsibilities reported. This flow, modeled in BPMN notation, describes the main stages involved from receiving the technical demand to validating the delivery, including activities such as feasibility analysis, defining the solution's architecture, aligning with stakeholders, documentation and monitoring implementation. The model seeks to represent in a visual and standardized way the recurring practices observed in the interviews, making it easier to understand the strategic and operational role of the Tech Lead in the solution development cycle.


Performace Tech Leader
Technical analysis
Receipt of new demand
Analyze the technical feasibility of the demand
Identify impacted systems and dependencies
Is the demand viable?
Yes
No
Communicate unfeasibility and propose alternatives (Back to PO)
Defining the solution
Define the technical architecture of the solution
Consult internal standards and restrictions
Validate alternatives (complexity, cost, time)
Yes
No
Needs external approval or validation
Submit technical proposal to superiors for validation
Alignment and communication
Align solution with PO and stakeholders
Support in prioritizing demand in the backlog
Translating technical implications into business language
Documentation and execution
Document technical decision
Monitoring the implementation of the solution
Do relevant changes arise during development?
Yes
No
Reassess technical impact and realign with PO
Validation and closure
Go to 'Evaluate results'
Record learnings or reusable patterns
Solution delivered with validated architecture

**Overlapping responsibilities between Product Owners and Tech Leads**

Although the Product Owner and Tech Lead play different roles in agile product development, the interviews revealed areas of overlap in their responsibilities. Both roles participate in refining demands, contribute to defining the scope of the solution and act as strategic interlocutors with stakeholders. Like POs, Tech Leads also participate in clarifying requirements, especially when dealing with technical restrictions and integration impacts. In many teams, Tech Leads actively contribute to discussions about the backlog, support prioritization decisions and are involved in translating business needs into technically viable solutions - responsibilities that, in theory, would be the domain of the PO. This intersection highlights the collaborative nature of demand refinement and reinforces the idea that product success depends on the synergy between commercial and technical leadership, but it can also generate a series of frictions that will be addressed later in this paper.

## 4.2 Types of Management Frameworks

In this section, the aim is to present the management frameworks used by company X in the daily processes of refining, organizing and structuring the activities carried out by the developers. The different approaches adopted will be explored and how each framework contributes to improving productivity, delivering value and adapting teams to project demands. Among the frameworks used are: Scrum, Kanban, Lean Software Development, Disciplined Agile, Scaled Agile Framework (SAFe), Large Scale Scrum (LeSS), Dynamic Systems Development Method (DSDM) and Lean Inception.

Scrum is one of the most widely used frameworks, organizing work in sprints with defined roles, such as Product Owner, Scrum Master and Development Team. It encourages collaboration and the continuous delivery of value. As Scrum.org points out, “Scrum is a simple framework for managing and doing work in teams” (Scrum.org, 2021). However, Scrum can be inflexible in environments with highly dynamic requirements and depends on the active presence of defined roles. To scale Scrum, frameworks such as Nexus are employed, extending Scrum to allow multiple teams to work on a single Product Backlog (Scrum.org, 2021).

Kanban, on the other hand, focuses on visualizing the workflow, limiting work in progress and maximizing the continuous delivery of value. It doesn't use sprints or fixed roles, offering flexibility for constant change. A 2020 study on Kanban highlights its effectiveness in improving workflow and reducing cycle time (Anderson, 2020). Kanban is ideal for teams that need flexibility and a continuous view of the workflow, but can be challenging for teams that need a more rigid structure.

Lean Software Development follows the principles of lean production, focusing on creating value and eliminating waste. According to a 2019 article, “Lean is about creating value for the customer while minimizing waste” (Poppendieck, 2019). Its implementation can be challenging in complex environments, but it is effective for improving processes and reducing costs.

Disciplined Agile (DA) integrates various agile practices, such as Scrum, Kanban and Lean, to cover the entire software development lifecycle. According to Scott W. Ambler and Mark Lines in their book Choose Your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working (2020), “DA allows teams to choose the right practices for their context” (Ambler & Lines, 2020). DA is ideal for organizations looking for a flexible and holistic approach, although its complexity can be challenging due to the integration of multiple practices.

Scaled Agile Framework (SAFe) was developed to help large organizations implement agile practices on a large scale, integrating Lean, Agile and DevOps concepts. Dean Leffingwell, in the book SAFe 5.0: The World's Leading Framework for Enterprise Agility (2020), states that “SAFe provides a framework for organizations to implement Agile at scale” (Leffingwell, 2020). Although effective, its implementation can be complex and require a significant investment in training and certification.

Large Scale Scrum (LeSS) is a minimalist approach to scaling Scrum while maintaining its basic principles. Craig Larman and Bas Vodde, in the book Large-Scale Scrum: More with LeSS (2016), point out that “LeSS is simple and flexible, allowing organizations to maintain the simplicity of Scrum at scale” (Larman & Vodde, 2016). It is ideal for organizations that want to scale without adding excessive complexity, but may not be suitable for very complex environments.

Dynamic Systems Development Method (DSDM) is an agile framework that prioritizes the continuous delivery of value, with an emphasis on collaboration and risk management. Although there are few specific recent references to DSDM, it is recognized as an approach that focuses on customer satisfaction and proactive risk management, and is mentioned in several agile frameworks.

Lean Inception combines Lean principles with the inception phase of projects, focusing on clearly defining requirements and eliminating waste right from the start. It is mentioned in agile project initiation contexts as an approach that helps to clearly define requirements from the start of the project.

In summary, each framework discussed has its own particularities and contributes differently to Company X's development process. While Scrum offers structure and a focus on collaboration, Kanban provides flexibility and continuous visualization of the workflow. Lean Software Development prioritizes the elimination of waste, and Disciplined Agile allows for a personalized and holistic approach. Frameworks such as SAFe and LeSS help scale agile practices in large organizations, while DSDM and Lean Inception offer approaches focused on the continuous delivery of value and clear definition of requirements. Combining these frameworks allows the company to adapt to the needs of the project, maximizing productivity and value delivery.

### 4.2.1 Agile Frameworks and Organizational Roles

The agile frameworks that were outlined define different approaches to the organization of work, including specific roles that ensure the continuous and efficient delivery of value. In this section, we explore how the roles of Product Owner (PO), Solution Architect, and Tech Lead are distributed in the main frameworks used by Company X.

#### Scrum

- **Product Owner (PO):** Responsible for the Product Backlog, defining priorities and ensuring that the development team works on the items of greatest value to the business. Acts as the main point of contact between stakeholders and the development team.
- **Solution Architect:** Scrum does not formally define this role, but in companies that use it, the solution architect can act as a technical expert external to the

Scrum team, ensuring that architectural decisions are aligned with product requirements.

- **Tech Lead:** In Scrum, there is no official Tech Lead role, but this professional usually assumes a technical leadership position within the Development Team, ensuring good code practices, quality and technical alignment with the product vision.

## Kanban

- **Product Owner (PO):** There is no formal PO role in Kanban, but the work of prioritizing the backlog still takes place, often being distributed among the team or maintained by a product manager.
- **Solution Architect:** As Kanban is highly flexible, the architect's role varies according to the needs of the project. They can act as a technical consultant to ensure that ongoing changes do not negatively impact the system's architecture.
- **Tech Lead:** Works to continuously improve the workflow, ensuring that deliveries are efficient and that there are no technical bottlenecks in the development process.

## Lean Software Development

- **Product Owner (PO):** Emphasizes the continuous delivery of value and the elimination of waste. The PO here needs to be aligned with Lean principles, ensuring that only essential requirements are prioritized.
- **Solution Architect:** Works to eliminate waste in the architecture, promoting scalable and sustainable solutions, with a focus on simplicity and efficiency.
- **Tech Lead:** Leads the team in the use of Lean practices, ensuring that technical solutions are developed efficiently and without wasted effort.

## Disciplined Agile (DA)

- **Product Owner (PO):** In DA, this role can be more flexible, being called Product Manager at strategic levels. In addition to prioritizing the backlog, they can also manage the product portfolio.
- **Solution Architect:** Has an essential role, as DA emphasizes the need for a well-defined architecture to support different agile approaches. Can act both centrally and decentrally, depending on the context.

- **Tech Lead:** Responsible for ensuring the application of appropriate technical practices within the approach chosen by the team (Scrum, Kanban, Lean, etc.), facilitating collaboration between developers.

## Scaled Agile Framework (SAFe)

- **Product Owner (PO):** In SAFe, there are both Product Owners and Product Managers. The PO works within the agile team, refining the team's backlog and ensuring alignment with the Product Manager, who has a broader view of the product strategy.
- **Solution Architect:** In SAFe, the role of Solution Architect/Engineer is formalized, ensuring that the architecture is well planned to support large-scale business requirements. They work closely with System Architects and Enterprise Architects.
- **Tech Lead:** Generally referred to as System Team Lead, they work on the technical side within agile teams, ensuring the correct implementation of architectural solutions and facilitating integration between teams.

## Large Scale Scrum (LeSS)

- **Product Owner (PO):** In LeSS, there is only one PO for all Scrum teams, ensuring alignment and centralized prioritization for the entire organization.
- **Solution Architect:** This role is not formally defined in LeSS, as the framework seeks to keep Scrum as pure as possible. However, architects can act as facilitators to help teams make decentralized technical decisions.
- **Tech Lead:** As in Scrum, there is no formal Tech Lead role, but usually a more experienced developer takes on this role to ensure good practices and code quality.

## Dynamic Systems Development Method (DSDM)

- **Product Owner (PO):** In DSDM, this role is called Business Visionary or Business Ambassador, ensuring that the business requirements are understood and implemented correctly.
- **Solution Architect:** Acts as Technical Coordinator, ensuring that technical and architectural decisions are aligned with the project's objectives.
- **Tech Lead:** Works to ensure the application of good technical practices and to optimize development efficiency within the agile team.

### Lean Inception

- **Product Owner (PO):** Plays a fundamental role in the initial phase of the project, ensuring that the requirements and objectives are well defined before development begins.
- **Solution Architect:** Works with the team to ensure that the initial product definition takes important technical aspects into account.
- **Tech Lead:** Contributes to strategic technical decisions to avoid rework in the future, helping to establish a solid foundation for development.

## 4.3. Types of Solution Architectures

Solution Architecture is the discipline responsible for designing and implementing technological strategies to meet an organization's business needs. According to The Open Group (2018), “solution architecture defines the structure and behaviour of applications, data and technological infrastructure, ensuring that they are aligned with organizational goals”.

The architectures considered were designed based on the context in which “company X” operates and the way it carries out its processes. This approach ensures that the proposed solutions are aligned with the company's operational reality, addressing specific challenges and maximizing the effectiveness of technological implementations.

Solution Architecture must have certain characteristics, such as a focus on the business, which, according to Ross, Weill and Robertson (2006), means ensuring that the technologies adopted contribute directly to the company's strategic objectives, avoiding isolated solutions that are not aligned with the business. Strategic planning is also important because, according to Bass, Clements and Kazman (2012), a good architecture allows organizations to identify weaknesses and opportunities in the current infrastructure, ensuring that new solutions are scalable and sustainable. In addition, Solution Architecture must guarantee technological integration, as Lankhorst (2017) points out, ensuring that different systems can communicate efficiently, promoting interoperability and reducing organizational silos.

The following section will present the solution architectures that support the proposed technological strategies.

## SOA - Service-Oriented Architecture

Service Oriented Architecture (SOA) is a software design paradigm that aims to facilitate the construction of scalable and flexible systems, allowing business functionalities to be made available as reusable and interoperable services. SOA is based on three fundamental principles:

- **Modularity and Component Reuse:** Systems are divided into independent services, which can be developed, deployed and maintained efficiently, avoiding duplication of effort and facilitating system evolution.
- **Interoperability and Communication between Services:** Services communicate through standardized interfaces, such as APIs or web services, allowing different applications to consume them regardless of the technology used.
- **Abstraction and Encapsulation of Functionalities:** Each service encapsulates a specific functionality, hiding implementation details and exposing only the interface necessary for its use, making it easier to maintain and evolve the system.

## Architecture Based on Microservices

Microservice-based architecture is a style of software design that divides a system into a collection of small, independent services, each responsible for a specific functionality. These services communicate via well-defined interfaces, usually APIs, and are designed to be scalable, flexible and easy to maintain.

- **Independence and Loose Coupling:** Each microservice is developed, deployed and maintained independently, with communication via APIs, which allows for loose coupling between services.
- **Scalability and Flexibility:** Microservices can be scaled individually, which facilitates resource management and improves the efficiency of the system as a whole.
- **Fault tolerance:** If one service fails, it doesn't affect the others, allowing the system to continue working even in the event of partial failures.

- **Rapid Development and Deployment:** Teams can work in parallel on different services, speeding up the development and deployment cycle.

## Monolithic Architecture

Monolithic architecture is a traditional style of software design where all the functionalities of an application are integrated into a single unit. This means that the user interface, business logic and data access are combined into a single executable program. Monolithic architecture is often used in smaller projects or startups, due to its simplicity and ease of development.

- **Single Deployment Unit:** The entire system is deployed as a single unit, which makes initial deployment easier, but can be complicated for upgrades, as it requires re-deploying the entire application.
- **Tight Coupling:** The different components of the system are tightly coupled, which can make it difficult to modify and maintain the system over time.
- **Vertical Scalability:** Scalability is achieved by increasing the capacity of the servers where the application is hosted, which can be limited and expensive for large-scale systems.

## Event-Driven Architecture

Event-driven architecture (EDA) is a software design model based on publishing, processing and reacting to events in real time. This approach allows systems to operate independently, reacting to events as they occur, without the need for direct and immediate interactions between services.

- **Events as the Central Point:** Events are the main focus of EDA. They represent significant facts that occur in the system, such as a purchase made or a change of state.
- **Asynchronous Communication:** Services communicate through events, which allows for loose coupling between them. This means that event producers don't know who the consumers are, and vice versa.
- **Reaction to Events:** Each service decides how to react to an event, or whether to react at all. This provides flexibility and autonomy for the services, allowing them to operate independently.

## Architecture in N Layers

The N-layer architecture is an extension of the layered architecture, where the system is organized into a variable number of layers, each with a specific responsibility. This approach allows for a more flexible and scalable structure than the traditional three-tier architecture, which only includes presentation, application and data layers.

- **Hierarchical Organization:** The layers are organized hierarchically, with each layer interacting only with adjacent layers, which facilitates maintenance and system modularity.
- **Separation of Responsibilities:** Each layer has a clearly defined function, such as presentation, business logic, data persistence, etc., which promotes separation of concerns and code reuse.
- **Flexibility and Scalability:** The N-layer architecture allows new layers to be added or removed as necessary, without affecting the rest of the system, which makes it easier to adapt to changes in business requirements.

## 4.3.1 Architectural Comparison Summary

Below is a summary comparison of different software architectures, highlighting their advantages, disadvantages and common use cases. The table condenses the key points, making it easy to quickly compare each approach.

| Architecture | Advantages | Disadvantages | Use Cases |
|---|---|---|---|
| **Service-Oriented Architecture (SOA)** | - Modularity and service reuse<br>- Easy integration with legacy systems<br>- Scalability and flexibility | - More complex to implement<br>- Requires governance to maintain standards and compliance | - Large corporations with multiple heterogeneous systems<br>- Systems requiring high interoperability |

| | | | |
|---|---|---|---|
| **Microservices Architecture** | - Independent development and deployment<br><br>- Better scalability and fault tolerance<br><br>- Easy adoption of different technologies | - More complex management and monitoring<br><br>- Requires a robust communication system (e.g., API Gateway, Service Mesh) | - Systems requiring high scalability<br><br>- Companies aiming for faster development and delivery cycles |
| **Monolithic Architecture** | - Simpler to develop and deploy<br><br>- Lower internal latency since all components are in the same environment | - Difficult to scale horizontally<br><br>- Complex to maintain in large projects | - Applications with few modules<br><br>- Small teams and short deadlines |
| **Event-Driven Architecture (EDA)** | - High scalability and flexibility<br><br>- Better decoupling between components | - More complex event handling<br><br>- Difficult to trace and debug | - Systems requiring high availability and resilience<br><br>- Real-time data processing |
| **N-Tier Architecture** | - Better code organization<br><br>- Easier to maintain and evolve | - Can introduce unnecessary latencies<br><br>- Harder to scale compared to microservices | - Traditional corporate applications<br><br>- Systems requiring a clear separation of responsibilities |

### 4.3.2 Methodology Comparison

As well as analyzing the advantages and disadvantages of each architecture, it is also essential to compare them on the basis of important technical criteria such as scalability, complexity and coupling. These factors directly influence the performance, maintainability and suitability of each approach for different business scenarios. The following table provides a structured comparison, helping to identify the best methodology for specific system requirements.

| **Methodology** | **Scalability** | **Complexity** | **Coupling** | **Best Use Case** |
|---|---|---|---|---|
| SOA | High | High | Low | Large corporations, system integration |
| Microservices | High | High | Very Low | Distributed systems, rapid scalability |
| Monolithic | Low | Low | High | Small applications |
| Event-Driven | Very High | High | Very Low | Asynchronous systems, real-time |
| N-Tier | Medium | Medium | Medium | Enterprise systems |

Choosing the right methodology for architecting solutions depends on factors such as scalability, system complexity, integration needs and business model. Modern architectures, such as microservices and event-driven, are more suitable for distributed and highly scalable systems, while simpler solutions can benefit from a monolithic or N-layer approach. The correct definition has a direct impact on the efficiency and success of the solution developed.

## 4.4 Recording Information

The information contained in this section was drawn from the preliminary research in topics 4.1.1 and 4.1.2, as well as observational analysis of daily life within company X. Within this context, project-related information is recorded using three different platforms, each with its own specific purpose and function. Below is a description of how these platforms are used to ensure proper storage and access to information throughout the project cycle.

The first tool is Monday, which is used as the main platform for organizing demands and monitoring project progress. Within this tool, the “meeting notes” column is the responsibility of the PO (Product Owner), who adds information related to meetings, decisions and important points discussed during the course of projects. This approach centralizes the communication of meetings and makes it easier to monitor the progress of demands. However, its ability to store detailed information, such as flowcharts or code, is limited, as this tool is more geared towards managing tasks and not recording technical details.

The second is an internal environment called Wiki, which is used to store more detailed and structured information, such as explanatory texts, codes, links, flowcharts and any other type of technical content that requires more in-depth documentation. This platform allows content to be easily shared via links, which facilitates collaboration between teams from different areas. Organizing and structuring the content on the Wiki is important to ensure that the information is accessible and can be found easily by all team members.

The third and final is Azure DevOps tool that is used to record smaller, more specific tasks within projects. Each task contains a detailed description with flows and steps, allowing the team to track the progress and execution of activities. Although it's a good tool for managing the progress of tasks, adding information directly to task descriptions can become a bit scattered over time, especially in larger projects. It is therefore important to be clear about the inclusion of information to avoid essential data becoming scattered or difficult to locate.

Using different platforms to record information can generate some advantages, such as specialized data storage in each tool, but it also presents challenges, such as the risk of redundancy and the dispersion of information. To ensure the effectiveness of this process, it is important that the team follows a clear plan on where and how each type of information should be recorded. In addition, integrating these platforms can improve the flow of information and facilitate access to essential data during the project lifecycle.

# 5. Literature survey

The academic study of the roles and responsibilities of those involved in the architecture of agile solutions and methodologies relies on various academic and professional contributions. This section presents the main authors and their approaches within the academic context, providing a relevant theoretical overview for the comparative analysis of different roles and their respective responsibilities. This theoretical survey is essential to support the proposal of a framework for Product Owners, allowing the integration of agile practices and solution architecture from an academic perspective.

To ensure a comprehensive and systematic approach to the literature review, we adopted a structured strategy to identify and analyze relevant academic and professional sources. The aim was to map the current landscape of roles and responsibilities in agile methodologies, with a particular focus on Product Owners (POs), Tech Leads and Solution Architects. The search was guided by clear inclusion criteria: renowned academic articles and reports published in the last 10 years, although we did find articles related to these authors and in these readings that helped us to better understand our research. Platforms such as Google Scholar and Scopus were prioritized for their reliability and breadth of coverage. Keywords such as “responsibilities”, “roles”, “agile methodologies”, “Product Owner”, “Tech Lead” and “Solution Architecture” were used to create search strings, ensuring a focused but comprehensive retrieval of relevant material.

The analysis was organized into thematic categories, including specific roles (e.g. Scrum Master, PO), core responsibilities, common challenges and emerging best practices. This categorization allowed for a clear comparison of different approaches and facilitated the identification of gaps and trends in the literature. This approach - combining academic rigor with real-world knowledge - ensured a balanced perspective on the topic.

Finally, the preliminary conclusions were consolidated to inform the development of the frameworks scenario. By comparing the practices identified with the specific challenges faced at the institution, we were able to visualize the real needs and were able to see in some ways that they are expressed in the frameworks, thus aligning the reality of the institution with established academic norms.This structured methodology not only increases the credibility of our research, but also allows us to explore a clear roadmap for integrating theoretical knowledge into practical solutions.

### 5.1.1. Roles in Solution Architecture

Solution architecture involves multiple players, from Product Owners to software architects and technology managers. In the academic context, the following studies stand out:

Parducci and Oliveira (2013), in the book TOGAF: IT Solutions Architecture for Enterprises, address how structuring enterprise architecture can improve organizational efficiency. The TOGAF framework defines clear roles for solution architects and enterprise architects but lacks specific guidelines for Product Owners, a point widely explored in academic research.

The Open Group - Responsible for developing TOGAF, one of the most widely adopted frameworks for enterprise architecture and solutions, defines roles such as Architecture Board and Solution Architects, but without a specific academic approach to the autonomy of POs in defining architecture.

### 5.1.2 Scrum and the Distribution of Responsibilities

Scrum is one of the most studied agile methodologies in academia, with several scientific articles exploring the clear definition of the roles involved:

Schwaber (2020), one of the creators of Scrum, points out that the Product Owner is responsible for maximizing the value of the product, but his influence on the architecture of the solution depends on collaboration with the technical team. Academic research indicates that this influence can be more active than originally proposed by the framework.

Kawamoto (2017), in her thesis *Scrum-DR: An Extension of the Scrum Framework Adherent to the Capability Maturity Model Using Design Rationale Techniques*, adapts Scrum to integrate predictability and stability into CMMI-DEV, indicating that the PO can be more involved in defining architectural requirements, a frequent theme in academic studies on hybrid methodologies.

Garcia (2015) adapted Scrum for different projects, pointing out that responsibility for the architecture tends to be concentrated on the developers, with little direct involvement from the Product Owners, a challenge identified by academic research into agile governance.

### 5.1.3 Kanban and Architecture Management

The Kanban method has been widely studied in the academic context for its applicability to agile project management:

Anderson (2020) defines Kanban as an evolutionary system that allows incremental changes to the architecture without a rigid definition of roles, which can make it difficult for POs to actively participate in the solution architecture, an aspect analyzed by academic studies on decentralized governance.

Ōno (1997), creator of Kanban at Toyota, focuses on continuous improvement, but does not specify how the roles relate to the solution architecture, a gap often discussed in academic research on the application of Kanban in software.

Bozheva and Jones (2021), co-authors of the *Kanban Maturity Model*, highlight the need for cultural evolution in organizations so that different profiles, including Product Owners, can contribute to architectural decisions, a recurring theme in academic literature on change management in IT.

### 5.1.4 Lean Software Development and Shared Responsibility

The Lean approach has been widely explored in academic research for its contribution to optimizing software development:

Poppendieck and Poppendieck (2019) introduce Lean principles applied to software development, arguing that architectural decisions should be made in a decentralized manner, involving Product Owners, developers, and architects. This approach has been validated by academic studies on large-scale agile development.

Beck (2004), creator of Extreme Programming (XP), argues that architecture should emerge from interactions between teams, suggesting that Product Owners can participate in decisions as long as they have sufficient technical knowledge. Academic research reinforces this view by analyzing collaboration models between POs and architects.

## 5.1.5 Agile Frameworks and Roles in Architecture

The evolution of agile methodologies has resulted in hybrid frameworks that combine different approaches to defining roles and responsibilities and are widely discussed in academic circles:

Ambler and Lines (2020), creators of the Disciplined Agile Framework (DAF), propose a model that incorporates elements of Scrum, Kanban, and Lean, allowing greater flexibility in defining roles. However, the DAF still keeps the architecture of solutions under the responsibility of software architects, a point often questioned in academic studies on PO autonomy.

Larman and Vodde (2016), creators of Large Scale Scrum (LeSS), emphasize the need to align architecture with business strategy but suggest that Product Owners should focus on prioritizing value, leaving architectural decisions to technical experts—an approach that has been critically analyzed in several academic publications.

A review of the academic literature shows that different agile and solution architecture approaches have significant variations in the definition of the roles and responsibilities of those involved. While frameworks such as TOGAF and DAF keep the architecture under the responsibility of technical specialists, approaches such as Lean and XP propose greater decentralization, allowing the active participation of Product Owners in the architecture of solutions, a recurring theme in academic research on agile governance.

Based on this academic analysis, the development of a framework for Product Owners should consider a balance between autonomy and technical support, allowing these professionals to make architectural decisions without compromising the technical coherence of the solutions. The next section of the paper will explore how to structure this framework to guarantee scalability, continuity and governance in the management of critical systems, always from an academic perspective based on scientific research.

# 6. Conclusion

The first module of this project focused on understanding the context, limitations and opportunities related to the participation of Product Owners (POs) in defining the solution architecture. Through a progressive approach over five sprints, it was possible to build a foundation for the development of the proposed structure.

Initially, the project documentation was drawn up and the TAPI was established, formalizing the objectives, scope and methods to be used. Next, the approach to the literature review was defined, with the identification of relevant databases and criteria for selecting sources, enabling a survey of existing theoretical references and frameworks related to solution architecture and the role of OPs.

The third stage focused on preliminary analysis of the literature and benchmarking of market practices. This phase revealed the absence of consolidated models that strategically integrate the role of POs in architecture, highlighting the gap addressed by this project. Practices adopted in other organizations were also identified which could serve as inspiration for the construction of the proposed framework.

In the fourth sprint, interviews were conducted with Product Owners, Tech Leads, Solution Architects and Developers, allowing for a deeper understanding of the challenges faced in practice. The interviews provided valuable insights into the current dynamics of the teams, the main points of friction between business and technology, and the perceived limitations of the POs' autonomy in structuring solutions.

Finally, the data obtained was consolidated and began to be translated into representative flows of the current process (such as BPMN diagrams), allowing visualization of the stages involved from the arrival of a demand to the completion of development. Adjustments were also made to the document's structure and a roadmap for the next project modules was established, based on the lessons learned so far.

Module 2 will build on this input, focusing on analyzing the current, ideal and desired flows in the architecture definition process. From this, the main components of the proposed framework will be identified, such as the responsibility matrix, documentation templates and support tools. The module will include the creation of templates and practical guidelines, the choice of modeling tools and the structuring of a simplified flow for using the framework. At the end, an MVP and effectiveness criteria will be defined, preparing the ground for the practical testing phase with Product Owners.

# 7. Bibliographical references

ABNT NBR 6023:2018. Informação e documentação: Referências: Elaboração. Rio de Janeiro: ABNT, 2018.

AMBLER, S. W., & LINES, M. (2020). Choose Your WoW!: A Disciplined Agile Delivery Handbook for Optimizing Your Way of Working.

ANDERSON, D. J. (2020). Kanban: Evolutionary Change for Your Technology Business (Edição atualizada).

BASS, L.; CLEMENTS, P.; KAZMAN, R.. Software Architecture in Practice. Addison-Wesley, 2012.

BECK, Kent. Extreme Programming Explained: Embrace Change. 2. ed. Boston: Addison-Wesley, 2004.

BOZHEVA, Theodora; JONES, David. Kanban Maturity Model: A Map to Organizational Agility, Resilience, and Reinvention. 2. ed. Sequim: Lean Kanban University Press, 2021.

Brydon-Miller, M., Greenwood, D., & Maguire, P. (2019). Why action research? Action Research, 17(1), 3-10.

Coghlan, D., & Shani, A. B. (2018). Conducting Action Research for Business and Management Students. Sage Publications.

COHN, M.. Agile Estimating and Planning. Addison-Wesley, 2019.

DIAS, Priscila Franco; OLIVEIRA, Guilherme Saramago de; GUIMARÃES, Iara Vieira. *Sinergia e desafios da pesquisa-ação no contexto educacional: caminhos para reflexão e transformação profissional*. Humanidades & Tecnologia (FINOM), v. 46, p. 24-30, jan./mar. 2024. Disponível em: DOI: 10.5281/zenodo.10565138.

ESCOLA DISRUPTIVAS. Benchmarking Acadêmico. Disponível em: https://escolasdisruptivas.com.br/glossario/benchmarking-academico/. Acesso em: 20 fev. 2025.

GARCIA, Luciano Anisio. Adaptação do Scrum para Diferentes Tipos de Projetos. Dissertação (Mestrado) — Universidade Federal de Pernambuco, Recife, 2015. Disponível em: https://www.agileleanhouse.com/lib/lib/People/KenSchwaber/Agile%20Project%20Management%20With%20Scrum%20-www.itworkss.com.pdf. Acesso em: 11 mar. 2025.

GREGORY, P.; MILLER, J.; MULLER, M.. Agile Project Management with Scrum. IEEE Software, v. 38, n. 2, p. 18-25, 2021.

Greenwood, D. (2022). Action research and organizational development: Bridging theory and practice. Action Research Journal, 20(2), 123-137.

HODA, R.; NOBLE, J.. Self-Organizing Roles on Agile Software Development Teams. IEEE Software, v. 34, n. 4, p. 44-51, 2017.

HOFFMANN, Valmir Emil; FARIAS, Juliano Silva. Saturação teórica em pesquisas qualitativas: relato de uma experiência de aplicação em estudo na área de administração. Revista de Ciências da Administração, v. 20, n. 52, p. 40-53, dez. 2018.

KAWAMOTO, Sandra. Scrum-DR: An Extension of the Scrum Framework Adherent to the Capability Maturity Model Using Design Rationale Techniques. Dissertação (Mestrado) — Universidade de São Paulo, São Paulo, 2017. Disponível em: https://www.teses.usp.br/teses/disponiveis/3/3141/tde-12032018-153055/publico/SandraKawamotoCorr17.pdf. Acesso em 11 mar. 2025.

LANKHORST, M.. Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, 2017.

LARMAN, C., & VODDE, B. (2016). Large-Scale Scrum: More with LeSS.

LEFFINGWELL, D. (2011). Scaling Software Agility: Best Practices for Large Enterprises. Addison-Wesley.

LEFFINGWELL, D. (2020). SAFe 5.0: The World’s Leading Framework for Enterprise Agility.

LEWIN, K. (1946). Action research and minority problems. Journal of Social Issues, 2(4), 34-46. Disponível em: https://spssi.onlinelibrary.wiley.com/doi/10.1111/j.1540-4560.1946.tb02295.x. Acesso em: 11 mar. 2025.

Mc NIFF, J. (2017). Action Research: All You Need to Know. Sage Publications.

MENON, A.; SINHA, R.; MACDONELL, S.. Stakeholder Engagement in Agile Projects: The Role of the Solution Architect. Journal of Systems and Software, v. 179, 2021.

MOE, N. B.; ÅGERFALK, P. J.; CONBOY, K.. Agile Processes in Software Engineering and Extreme Programming: 20th International Conference, XP 2019, Montreal, Canada, May 21–25, 2019, Proceedings. Springer, 2019.

OLIVEIRA, J.; SILVA, R. Gestão Estratégica: Ferramentas para Competitividade. São Paulo: Atlas, 2018.

ŌNO, Taiichi. O Sistema Toyota de Produção: Além da Produção em Massa. Porto Alegre: Bookman, 1997.

PARDUCCI, Renato Jardim; OLIVEIRA, Elisamara de. TOGAF: Arquitetura de Soluções de TI para Empresas. São Paulo: Phorte Editora, 2013.

PICHLER, R. (2010). Agile Product Management with Scrum: Creating Products that Customers Love. Addison-Wesley.

PICHLER, R. (2018). Agile Product Management with Scrum: Creating Products That Customers Love. Addison-Wesley.

POPPENDIECK, M. (2019). Lean Software Development: An Agile Toolkit (Edição revisada).

RADIGAN, D. (2018). Product Roadmaps Relaunched: How to Set Direction while Embracing Uncertainty. O'Reilly Media.

ROSS, J. W.; WEILL, P.; ROBERTSON, D. C. Enterprise Architecture as Strategy: Creating a Foundation for Business Execution. Harvard Business School Press, 2006.

SCHWABER, K.; SUTHERLAND, J.. Guia do Scrum. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-European.pdf. Acesso em: 20 fev. 2025.

SCHWABER, K. Agile Project Management with Scrum. Redmond, WA: Microsoft Press, 2004. Disponível em: https://www.agileleanhouse.com/lib/lib/People/KenSchwaber/Agile%20Project%20Management%20With%20Scrum%20-www.itworkss.com.pdf. Acesso em 11 mar. 2025.

SMITH, J. (2021). Agile Tools for Modern Product Management. Journal of Agile Practices, 12(3), 45–60.

SVERRISDOTTIR, H. S.; INGASON, H. T.; JONASSON, H. I. The role of the product owner in Scrum – Comparison between theory and practices. Procedia - Social and Behavioral Sciences, v. 119, p. 257–267, 2014.

THE OPEN GROUP. TOGAF Version 9.1. 2018. Disponível em: https://pubs.opengroup.org/architecture/togaf91-doc/arch/chap52.html. Acesso em: 20 fev. 2025.

UNISINOS. Tech Lead: Papel e Responsabilidades. 2024. Disponível em: https://ead.unisinos.br/tech-lead. Acesso em: 20 fev. 2025.

# Appendix

## Appendix  A - POs survey answers

### Survey 1 - 17/02 to 28/02

**How do requests reach you?**
*(Describe where the requests come from, who participates in this process and how the demands are assigned)*

**Person 1 :** Ideally, they would arrive via a list of priorities classified in numerical order, but this didn't happen in practice, so armed with this material, every quarter I would pass on what I understood from internal conversations about what the priorities were, based on the company's strategy, and from this we would do a pre-refinement, classifying the size of the effort and its possible dependencies on other teams. Before the PI, we formalized it via email, understood whether our enablers would also be able to tackle these priorities and, if necessary, reallocated the priorities. By the time the PI took place, everything had already been defined, everyone involved was aligned and all we had to do was formalize the demands that would be worked on during the quarter and what the expected results were, answering what each feature would solve and what the expected result would be.

**Person 2:** Mostly from the Business/Commercial team.

**Person 3:** Ideally, the main requests are defined before the start of the year, according to the strategic planning of each product for the year, and broken down in a macro way in the quarters. This way, we already have a forecast of the items and at each sprint review we review the prioritization of demands in a more micro way. However, business stakeholders often have “off-plan” requests, mainly from commercial contacts in which customers report problems or opportunities for improvement. Also, as the competition moves, the business team ends up needing to launch new features/products to keep up with the market, which often leads to replanning. The demands usually come from the business team, but there are also technical demands that the IT team itself prioritizes. Also, partner areas such as customer service, marketing, operations and UX may have unplanned requests that need to be prioritized. The assignment of demands takes place within each squad. Each developer/designer has a few products of greater expertise, and the demands are divided between the focal points of each product, according to their delivery capacity and the complexity/effort of each demand.

**Person 4:** The demands are defined by the business team

**Person 5:** As a rule, the demands I work on come through oral requests or messages/emails sent by my manager.

**When a demand comes in, is it completely defined or do you need to refine it?**
(Explain whether demands come ready-made or whether there is room for improvement of the problem)

**Person 1:** I don't remember fully defined demands. In refinement, you explore scenarios that were not clear without this step. This is the time to explore success and failure scenarios and, above all, to understand whether the solution is just for the team to build or whether it has dependencies.

**Person 2:** Mostly refinement is needed.

**Person 3:** Most of the time, there is plenty of room for refinement. The more complex the demand, the more refinement time is needed. Only minor adjustments/improvements come with complete or almost complete understanding. It usually takes one or several calls of understanding before we arrive at the desired delivery properly refined.

**Person 4:** Needs refining, from understanding the business requirements, what the business team and the client expect, to refining technical requirements with the development team or other areas involved.

**Person 5:** Most of the time, the demands arrive ready-made, but in some specific cases, they need to be improved and a greater understanding of the objective and the needs to be met is required.

**How do you refine demands on a daily basis?**
(Describe your refinement process, what steps you follow, who participates and what tools or methods you use).

**Person 1:** For the planning of the quarter, we had to wait for confirmation of this list of priorities, which happened about 15 days before PI Planning, but throughout the quarter, we had a recurring schedule of weekly refinements with the teams. At this meeting, we looked at the health of the development that was underway, understanding whether the schedule that had been drawn up when the topic was prioritized was adhering to expectations and also, when possible, taking the opportunity to pull out the increments of our deliveries, for example, in one quarter we were working on an MVP, what the next step was and what to expect from this increment. We also looked at the demands in the prioritization funnel that hadn't been prioritized previously due to lack of capacity. The risk in this second example is that, over the course of the quarter, this backlog loses priority due to any other internal or external factor, such as a regulatory demand, for example.

**Person 2:** I start by calling in the teams that will provide the input prior to IT, such as Legal for legal validation of what we intend to develop, Fraud to check that the product we are thinking of won't generate security gaps, Finance, Tax, Commissioning (if the product is sold by partners), Billing if the bank chooses to charge the customer fees for using it, always accompanied by the business team. At each meeting, I document what has been agreed in a WIKI and formalize it via email. From then on, I start drawing in BPMN to show what the customer journey will look like and which internal systems will be involved. teams, etc. Usually in the midst of refinement, new needs arise and new teams are activated. Once we have reached a consensus and everyone knows which activities they will be responsible for, I use monday to manage progress and control dates.

**Person 3:** Refinement always involves the requesting business stakeholder, to explain the need and expected outcome, as well as defining the business rules. Also, the technical lead responsible for the product, to understand the technical needs, solution possibilities and effort involved. Ideally, we also involve someone from UX as soon as possible, so that they are already aware of the needs. The process involves documenting the understanding/rules, as well as designing the flows involved and mapping the people who need to be involved/areas that will be impacted. Example: will there be an impact on security flows? if so, involve the security squad; does this feature/product involve partner remuneration? if so, activate the fee/remuneration team; and so on.

**Person 4:** Business alignment meeting, technical refinement, meeting with areas involved and planned delivery

**Person 5:** Refinement is usually done by contacting stakeholders. When this request is being made to someone from another team, I usually contact the person to better understand what their needs are. When the person in question is my manager, I talk to him in parallel to better understand what he wants done.

**Do you feel you have enough autonomy to refine the demands in the best way? Why?** (Explain whether you have the freedom to adjust demands as necessary or whether there are restrictions that hinder this process).

**Person 1:** Sometimes you won't have one and your role is to try to involve those people who do, for example, the business people. It's very important to make them feel that they are an integral part of that development and what their role and expected responsibility is. I've always had the practice of defining the first answers to some key questions in order to continue with the refinements with the teams: What are we doing? Why are we doing it now? What do we expect from this delivery? The answers to these questions make the refinement process easier, the team understands what the problem is to be solved and what to expect, with this we gain engagement and they must come from the business based on aligning a customer pain with the company's strategy. Another very important point is that these questions facilitate the reprioritization process, when a demand in the middle of the quartile comes with a request for prioritization and you have to reprioritize it together with the team. This is the role of the business and you as PO or PM can help by making the impacts clear, for example, will this new demand impact the backlog that expects to deliver these results, does this new demand deliver more value than those prioritized?

**Person 2:** Refinement is joint, but it's up to the PO to coordinate the whole process and keep everyone aware of what we're agreeing. So the process is not dictated by the PO, but flows as the meetings go by and we find new needs. One factor that can limit the work is the regulatory deadline, in which case there is no room for negotiating deadlines or phasing.

**Person 3:** It depends on the product/stakeholder. There are some that are already more defined (some stakeholders even make a prototype in the PPT), others that are super raw and the PO ends up having more autonomy and protagonism. During the process, if it is necessary to suggest/adjust the demand, I believe that with a good basis, the PO does have the autonomy to adjust the route. It's very important here to provide the reasons for changing the route. For example: suggesting breaking the delivery down into phases, so as to deliver value over time and iterate/validate hypotheses before arriving at the final delivery (which is often the business team's idea of the ideal, but not necessarily what the client needs).

**Person 4:** Yes and no, it depends on the type of demand, if it's something very specific to the client there's not much room for change, but if it's not, I'm free to give my opinion, change and even change the delivery priorities.

**Person 5:** In cases where it's necessary to refine the demand, I feel I have the necessary freedom, but I'm not confident enough to go along with my opinion. Therefore, I always contact my manager to align expectations and ensure that what I have imagined/understood is exactly what he wants done.

**What are the main challenges you face in refining demands?**

(List the main obstacles that hinder more efficient refinement, such as lack of information, time constraints, lack of alignment with stakeholders, etc.)

**Person 1:** When there are external dependencies on other teams. Everyone has their own priorities and sometimes they don't fit together. Demands without a defined scope of what is expected, so the team becomes demotivated and lost throughout the process. Lack of time aligned with the company's expectations. When it's the business that sets the deadline and not the team, and because of this lack of time, it's not possible to explore all possible scenarios. Team schedules: many times I've had to cancel a refinement due to a service in Production. In my old scenario, the delivery team was the same as the production team. A lot of changes in direction, which makes it difficult to focus. Several times I had to interrupt a refinement to meet a new demand or problem in Production and when I managed to reprioritize this discovery, the team had already lost the timeline and reasoning and we had to take a step back to understand the demand again, thus increasing the discovery time. Requests to start technical refinement without a prototype. This is a lot of rework, because putting together a service architecture without understanding what is expected to be delivered to the end user in most cases means putting effort into something that will probably need to be revisited.

**Person 2:** The biggest obstacle is the deadline expected by the business and the pressure generated to get everything out very quickly, which hinders the refinement that often continues to take place while development is underway.

**Person 3:** I believe that the root cause is always a lack of clarity about the OBJECTIVE of the delivery. What are we doing this for? What is the expected result at the end? I often feel that we are doing and doing, and we don't have a clear objective. Once the objective is clear, we can suggest other ways of getting there. I feel like we're working on the WHAT and not the WHY? Having the WHY, we can redefine the HOW and have more autonomy. In addition, there is often a lack of information about the product, about the need, and time for refinement is tight because of the deadline set for delivery.

**Person 4:** Failure to align with everyone involved, asking for priority/urgency on a demand that the person doesn't even know what they want done.

**Person 5:** I believe that, as a rule, the main problem is fully understanding all the stages of the task before starting it. Most of the time, I only discover the complexity of the task once I've started it. Before that, it's very difficult to have that dimension and be able to work in the best way.

## Survey 2 - 17/03 a 28/03

**How do requests reach you?**
(Describe where the requests come from, who participates in this process and how the demands are assigned)

**Person 1:** It currently comes from the bank's internal areas and also from the leaders in my area

**Person 2:** They come via the client, usually at a meeting to discuss the project. But sometimes it comes from the leaders of the team that worked on it, like a top down.

**Person 3:** Through management tools with agile dashboards (Scrum and Kanban)

**Person 4:** The service areas are determined by the strategic leadership of the team and requests come directly from the customer, either by opening tickets, requesting a message on Teams or email or by actively prospecting for demands.

**Person 5:** Requests originate from emails, Whatsapp messages, work orders, calls or virtual meetings and generally come from customers. Sometimes, the request arises from a need identified by the team as a result of other demands. In the vast majority of cases, the demand is assigned by the team's techlead, which in our case can be three: backend, frontend or QA.

**Person 6:** I work as a PO in a software factory, where we deliver solutions to the client through commercial partners. Therefore, the demands come through the client's project management tool, Azure, in collaboration with the business partner's requirements team.

**When a demand comes in, is it completely defined or do you need to refine it?**
(Explain whether demands come ready-made or whether there is room for improvement of the problem)

**Person 1:** It's necessary to refine, usually we only know the central theme when it arrives, and then if we see that it's a good project, we book in to understand it in more detail.

**Person 2:** It depends, but for the most part it's necessary to refine, to go into the granular, to understand the business rules, the systems used and the context of the project. Few times does the client come with rounded information; when they did, it was using the AGNS specification and the project was small.

**Person 3:** Most of the time it needs to be refined

**Person 4:** It is necessary to refine the problem, design the sub-process affected by the requested demand and, occasionally, contact other teams related to the requesting client for technical and business alignment.

**Person 5:** They don't always arrive with a satisfactory level of detail. Sometimes we are free to improve the description of the requirement (when it's a simpler feature) and sometimes we meet with the client to get a better understanding. In the case of very large demands, the client is required to go into as much detail as possible, which often doesn't rule out the need for alignment meetings.

**Person 6:** The vast majority of requests come with a macro definition of the business rules and therefore require an in-depth study of the business rules and possible impacts on the system before the request is taken to the development team.

**How do you refine demands on a daily basis?**
*(Describe your refinement process, what steps you follow, who participates and what tools or methods you use).*

**Person 1:** We usually arrange a meeting with the owner of the routine to understand how it is currently done, so that we can understand in detail where we can act.

**Person 2:** Through meetings with clients, recorded, and groups/chats for questions during the specification. After these understandings, they are documented in the confluence.

**Person 3:** The refinement process usually takes place after the Daily meetings, in which the development tasks are discussed and the situation they are in. These meetings take place with the client.

**Person 4:** The specification process involves active or passive discovery (depending on the origin of the demand) of the entire macro-process directly with the client. Once all the materials have been collected, be they documents, videos, spreadsheets, presentations, etc., the entire process is documented, a flowchart drawn and a technical solution proposed. The correct cases and possible ramifications are mapped out to avoid errors.

**Person 5:** It depends on the complexity of the demand. If it's something simple, I myself access the application in which the improvement will be applied, understand the feasibility and register the demand in Jira, for later allocation of a dev to work on it. If the demand was more complex, it was necessary to meet with other business analysts (optional) and also with the techlead of the areas involved (essential). The construction of flowcharts is also a relevant activity in some cases. Some of our applications require prototyping in Figma. Others only require written guidelines or simple changes to the system's screenshot. These are attached to the demand card to guide the dev. There are cases in which, after detailing the card, the dev comes to us to reinforce the understanding of the activity for a pre-validation and we also have the SCRUM ceremonies, the sprint planning meeting is also an occasion to refine the understanding.

**Person 6:** We hold a business refinement meeting, where I make a point of taking part in all the requirements gathering agendas. This agenda is attended not only by me, but also by the commercial partner and the client. We currently use Scrum in the project, so the flow is as follows: we raise the priority demands with the client, we measure the effort and cost of development, we pass this information on to the client and the client tells us which demands we should develop first.

**Do you feel you have enough autonomy to refine the demands in the best way? Why?** *(Explain whether you have the freedom to adjust demands as necessary or whether there are restrictions that hinder this process).*

**Person 1:** No, currently to carry out more in-depth technical refinement, I turn to my technical partner to think about the architecture of the solution

**Person 2:** No, but because of the technical dependence on the protype team. I still need them to have conversations and understandings with the IT teams and also to think about solutions for the project.

**Person 3:** Yes. I always have a communication channel open with the client at any time to refine the demands.

**Person 4:** Yes, but the time available to consider a demand as refined often hinders the process of strategic planning and questioning that could lead to functionalities that are more adherent to a good product or sustainable solutions in the long term.

**Person 5:** We have freedom up to a point, when the demands don't involve a lot of work at backend level, integration, etc. What makes the process difficult is that we don't have the technical vision of the impact that certain requests will have on the applications and the functioning of the systems. That's why it's necessary to meet with the techleads to go over what has been demanded by the client and reach a common denominator and measure the effort of the deliveries.

**Person 6:** Yes. In order to make any adjustments to the demands, we depend on the client's acceptance and, in some cases, the commercial partner's approval.

**What are the main challenges you face in refining demands?**
*(List the main obstacles that hinder more efficient refinement, such as lack of information, time constraints, lack of alignment with stakeholders, etc.)*

**Person 1:** technical refinement and full understanding of the routine by the requesting area

**Person 2:** Getting the technical information from the IT team, getting to the detail of the information with the from/to filled in.

**Person 3:** The main problem is the lack of code documentation aligned with the business documentation to minimize verification time for a requirement or demand.

**Person 4:** Time constraints, lack of alignment with leadership on strategy and incomplete information, forcing direct refinement with the client to be prolonged.

**Person 5:** Lack of clear information from the client and even indecision about what they really need; time constraints, both because of possible client urgencies and because of the techleads' scarce time for alignment meetings; lack of knowledge of contractual details, because sometimes what the client wants will cost X hours worked and the client may not have this expected guarantee of payment; Little technical knowledge of the effort needed to implement a feature or suggest alternatives to it.

**Person 6:** Time constraints, sometimes we want to start developing the demand as soon as possible and we end up compromising an effective survey and refinement of the specifics of the demands.

## Appendix B - Tech leads and solution architects survey answers

### Survey 1 - 17/03 a 28/03

**How do technical demands reach you?** *Describe the process for receiving technical demands and how these demands are communicated to you (meetings, management platforms, direct communication with stakeholders, POs, etc.).*

**Person 1:** Demands come to me in 3 ways: from the PO responsible for the areas I serve, directly from clients in the areas of the bank I serve or have served and from my boss (usually internal demands from the area).

**Person 2:** Currently, requests come through a platform called Confluence, where the PO writes the specification of the problem we want to address. In addition, before the technical demands are ready, there are specification meetings with the client, in which some POs involve the tech lead responsible for the area or even the DEV who is interested or who will take on the demand.

**Person 3:** Alignment meetings with stakeholders

**Person 4:** Today there are stakeholders who are “owners of the business” and ask for something, as well as other peer leads who also ask, and there are also technical debts and today, as a business owner, we are also able to generate demands for ourselves, from large projects to small fixes

**Person 5:** Sometimes they come from the business team, when they see an opportunity that is worth exploring, other times from analysis and monitoring of metrics and system logs, which provide indications of the problems that users face on a day-to-day basis and, consequently, points for improvement in the system.

**Person 6:** In the last quarter, the technical demands came directly from contacting the client and specifying the routine. In the course of mapping the As Is, the requester comments that they need to consume data from X systems, so the next step is to seek out the person responsible for the systems and make an appointment with them to understand how to consume the information.

**How do you define and prioritize the technical approach for each demand you receive?** *What criteria do you use to decide on the best technical solution to meet the demand? How do you decide between different approaches or technologies?*

**Person 1:** I try to understand the problem well, look at the scalability required and possible changes or problems. Then I look at the solutions we've already developed and see how this new demand fits best with the technologies we've already used and that we know work well. If we haven't done anything like that yet, I try to search the internet for the best technology and solution to the problem.

**Person 2:** As our Automation area is a tactical area, we should prioritize an architecture for which we already have templates designed, with the aim of delivering quickly. Our solutions consist of lambdas and cronjobs, and we understand which one is the most suitable for what we are going to do. In addition, we evaluate the complexity of each demand to decide whether it is best to use a relational database or DynamoDB.

**Person 3:** It is analyzed whether the technical solution can scale beyond the basic scenarios envisaged, whether there is technological know-how within the resources and how complex maintenance is and what the cost is to keep the solution running.

**Person 4:** I think that in today's context we first evaluate a factor of complexityXtimeXvalue of delivery. Then we always design what we think is ideal, after understanding all the business needs and translating this into technology, what is the volume? criticality of the flow? can it be asynchronous? And according to the factor above, we understand what we can attack, try an mvp of the solution, or something like that

**Person 5:** Ease of maintenance of the planned solution, future scalability and assertiveness of the input data, to avoid miscalculations and possible losses for the client and the company.

**Person 6:** First, I understand the client's needs, considering the performance required and the method of consuming external data from other systems. I also take into account the need for exception flow mapping, understanding how the system needs to behave if any stage fails.

**Do you have the autonomy to define the architecture and technical solutions to the demands or are there limitations?** *Are there external factors (e.g. budget constraints, deadlines, stakeholder requirements) that affect your ability to define the ideal solution?*

**Person 1:** Most of the time I have the autonomy to decide, but I always end up choosing the quickest and most efficient solution, since I work on tactical projects that should be short term.

**Person 2:** I have the autonomy to design the technical solution we're going to tackle in the best way possible, but as we have delivery targets throughout the year, we have to prioritize agility and efficiency. That's why we rely heavily on solutions that have already been developed and that make our day-to-day work easier. With regard to external factors, we always have to be careful that the solutions we adopt are not excessively expensive if they are used inappropriately, such as the use of Glue Job, Athena and other tools. However, we have this initial control when defining the solution.

**Person 3:** Limited by budget

**Person 4:** There are certain limitations, budget is not so much a problem, but there are non-approved technologies and technologies that some higher tech stakeholders don't like, and this creates a taboo and an impossibility of being able to think autonomously about which technologies to use

**Person 5:** We have autonomy as long as the solution doesn't deviate from the design standards defined by the team, in order to guarantee easy learning by other colleagues if they need to work on our systems.

**Person 6:** I have a lot of autonomy to define the architecture and create proofs of concept of my ideas, but always with care for costs.

**What tools or methodologies do you use to coordinate the execution of technical demands with your team?** *Tell us about the tools such as Jira, Azure DevOps, Trello and others that you use to manage the technical execution of demands.*

**Person 1:** To keep track of my individual tasks I use OneNote, to write down everything I've talked about with the client and all the rationale for what I have to do on the project. To keep track of all the projects, the overall status and dates I use Azure DevOps.

**Person 2:** With regard to technical demands, we use Confluence for both the specification and the final design of the solution. In other words, once the DEV has finished, he has to document the code in Confluence. With regard to deadlines and tasks created, we use Azure DevOps for control. In addition, for my personal control over the developers, I use OneNote for general notes.

**Person 3:** Azure, Excel

**Person 4:** Monday and Azure Devops

**Person 5:** Monday to define the macro scope of activities, Azure DevOps to define the micro scope, including features and user histories, MS Teams/Outlook to control agendas and meetings. We follow some rites of the SCRUM methodology, such as daily meetings, planings and code reviews.

**Person 6:** I use Azure, creating tasks that should ideally last up to 2 days of development.

**What is the biggest difficulty you encounter when communicating technical needs and priorities to the development team?** *Are there any challenges in communicating clearly or setting expectations, both internally within the team and with other departments (POs, stakeholders, etc.)?*

**Person 1:** The biggest difficulty I experience is foreseeing all the technical problems that may arise or possible unforeseen integrations

**Person 2:** The biggest difficulty I encounter when communicating technical needs and priorities to the development team is mainly in integrations and the way data consumption is carried out. For example, if the consumption is via API, the estimated time is X; however, if it is via messaging, the time increases by 10 due to the complexity of saving the data and consuming it later. This difference in complexity is not always clear to everyone involved, especially stakeholders and POs, which can lead to challenges in communication and aligning expectations. When these integrations and complexities are already well defined in the initial project specification, I can better estimate deadlines and development effort.

**Person 3:** There can be noises in the understanding of demands in cases of alignment where the necessary people have not been involved.

**Person 4:** Today, as I sometimes end up playing the role of a stakeholder, not just a tech stakeholder, I find it a little difficult to change hats quickly, and this is also a little more burdensome than normal, and ends up making it difficult to write the tasks better, even when refining previous definitions

**Person 5:** The greatest difficulty lies in the technical understanding of the proposed solutions, which needs to be very well explained and aligned with all the teams involved in order to identify possible gaps in the systems and planned integrations. It is necessary for those involved to have a high technical level and to know their products and/or systems well so that the scopes are well defined, thus avoiding teams developing solutions that are already offered by other existing systems and known problems going unsolved.

**Person 6:** Stakeholders who don't understand the technical side of development often don't agree with certain deadlines. This alignment with clients is always a challenge.

**Have you ever faced challenges related to changing requirements or priorities during the development cycle?** *How do you deal with unexpected changes and how do they affect the proposed technical solution? How do you adjust the team's planning to adapt?*

**Person 1:** I've had to deal with scope changes on several projects, and they have a significant impact on the time it takes to deliver the solution. This often leads to frustration on the part of the client because they don't really understand that it wasn't mapped out, since it's very obvious to them certain steps in their daily lives. In these cases, I understand the new demand well, price this adjustment and pass it on to the team. I understand together with the team the development to be done, the stages and how long it would take

**Person 2:** Changes during development are quite common, often due to information that the client didn't include in the initial specification. This can cause anxiety among developers, but as a tech lead, we have to take a lot of the responsibility and come up with new approaches to deal with these situations. Generally, we increase the delivery time and assess what is feasible in the time available, leaving the rest for a version 2 of the project.

**Person 3:** Constant, the solutions are designed to be easily changed and evolved so that depending on the type of change, it shouldn't have such an impact on development time.

**Person 4:** It even happens frequently, we usually stop everything and replan as soon as possible, the fact that the development cycle is not so respected “helps” in this regard

**Person 5:** Yes, frequently, and this is part of the agile methodologies used by the team. When this happens, we bring the team together to discuss possible solutions and adjustments to the scope and find out who is responsible for the solution in the other teams involved in the project, so that we can then contact the right people, ensuring assertive and effective communications. Finally, we study possible extensions to pre-defined deadlines and align delivery date expectations to ensure that everyone is on the same page.

**Person 6:** Yes. The guideline is that if the proposed change prevents the use of the mapped solution in production, this point is formalized with the leaders of the requesting area and included in the delivery. Otherwise, the change is barred from the current delivery and allocated to the next version of the project.

**What is your collaboration process with POs and stakeholders throughout the development cycle?** *When do you interact with POs or stakeholders during the process? How do these interactions impact the definition and execution of technical solutions?*

**Person 1:** I interact with the PO from the moment he has understood the problem with the client and put together a specification, even if it's just a sketch. I help in the process with the technical parts, designing the solution and checking the feasibility of the solution.

**Person 2:** The collaboration process is usually quite effective. Both the PO and the stakeholders usually explain any business rules in detail or clarify any doubts with the developer responsible for the requirement, involving the IT teams when necessary. These interactions, depending on the case, can alter the initial scope of the project due to requested changes or improve initial understanding, which in turn speeds up delivery.

**Person 3:** Surveying the effort and tasks of a given demand, and also understanding the demand in full with the stakeholders.

**Person 4:** We interact a lot in refinements, it helps a lot, it ends up having an impact on how the task is going to be defined, even before the refinement, and it's essential, whenever this triad of tech, product and business happens, it's ideal

**Person 5:** Interaction with stakeholders happens all the time through project progress reports, release notes with the latest features and scheduled and spontaneous meetings to resolve doubts. This constant interaction is essential so that expectations are always aligned, those responsible are always well identified and any deviations are not taken by surprise, avoiding friction between the areas involved.

**Person 6:** Mainly in mapping the As Is and defining the screen prototype

# Appendix C - Developer survey answers

## Survey 1 - 17/02 to 28/02

1. How do work demands come to you on a daily basis?

(Describe the process, including where you receive it and how it is assigned).

Person 1: I have routine demands such as sending daily reports, and other operations that arrive by e-mail, Bloomberg terminal, automation project demands I usually ask if there is anything to do, and momentary demands usually an analyst comes to my desk and asks for it

Person 2: These demands come in two ways. Either from meetings I have with clients in my business area (the clients are internal), in which they request new features or projects, or from conversations with my boss, in which he does the same thing as above, only privately, without me. In short, it all comes down to talking to people in the business area, either myself or my boss, and passing it on to me. In the end, I usually record this myself on an Azure DevOps board, so that I have a clear flow of tasks, as well as development time and delivery deadlines.

Person 3: My manager currently has a conversation with the project's stakeholders and they define the scope to be worked on and the demands of that scope. After this conversation between them, my manager sets up a meeting with me and gives me the context of the demand, how I have to act on it and what the delivery deadlines are.

Person 4: Tech Lead

Person 5: Mainly at the request of the business team, but also by analyzing system logs and taking a critical look at the portal, which allows us to understand the customer journey and which points need improvement. Tasks are assigned by analyzing the workload of the team members (Azure's backlog helps a lot here) and each person's previous experience with the topic to be developed.

Person 6: Demand from stakeholders

Person 7: It can either come from the commercial or operational team, in some cases there may be projects that come from a partner, but a good part of it is also proactive, always taking into account the success of the products and the benefit for those impacted.

Person 8: By the squad leaders, usually the teams. It's usually passed on by them in the same way.

Person 9: The demands come mainly through conversations, without a centralized system to keep track of the tasks. When a new project starts, we hold a meeting to understand the requirements, and from there, I carry out the work until the final delivery, with no structured monitoring of progress.

Person 10: It usually happens on two fronts. The product team can segment some demands, either regulatory terms or business needs, or these demands can come from QA (some defect or fix needed), or from some Product Manager in the tech area. We usually receive these demands by e-mail, and the assignment is made by creating a JIRA by our VP (Lead) or a Senior and redirecting it to the DEVs.

Person 11: Arrives via PO and Tech Lead

2. Do you think that the way demands come to you currently works well or could it be improved?

- It works well, I don't see any need for change.
- It works, but it could be better.
- It doesn't work well, it needs improvement.
- Others...

Person 1: It works, but it could be better.

Person 2: It works, but it could be better.

Person 3: It doesn't work well, it needs improvement.

Person 4: It works well, I don't see any need for change.

Person 5: Other: Works well for the size of the team and scope of the product. Excessive control can be detrimental and lead to micromanagement and a consequent decrease in morale.

Person 6: Works well, no need for change.

Person 7: It works, but it could be better.

Person 8: It doesn't work well, it needs improvement.

Person 9: It works, but it could be better.

Person 10: Works, but could be better.

Person 11: Works, but could be better.

3. If you think it could be improved, how could this process be made more efficient?

Person 1: A list of tasks and pending items, with priorities

Person 2: It would be ideal to have a system capable of listening to the meetings, automatically organizing the ideas discussed and mapping out new flows requested. At the end of the meeting, the developers or tech leads could review, edit and approve the suggestions. Once approved, the system would integrate the flows directly into management tools such as Azure DevOps, Monday, Jira, Trello, among others.

Person 3: I think that if I had direct contact with the stakeholders I could have a better contextualization of the problem and how the solution applies to it, as well as being able to clear up any doubts and propose improvements to what has been worked on, giving me more prominence within the work rather than just being a conveyor belt for producing demands.

Person 4: No answer

Person 5: No answer

Person 6: No answer

Person 7: I actively participate in business decisions to evolve the products and, to a certain extent, the software I work with, which I believe evolves well without deviating from the ideal strategy, so demands are 'micro-managed' with care, but I believe that urgent demands with a general impact on a larger scope should be passed through a committee of technical managers to assess the general risks and effectively plan the implementation of these changes. Today the teams act individually and decentrally.

Person 8: I believe that better organizational methods could be used. Using agile methods and tools to organize these activities, prioritizing the really urgent ones.

Person 9: The process would be more efficient with a tool to record and monitor demands. This would help to better organize the work, give more visibility to the progress of tasks and improve communication.

Person 10: The scope of the change could be a little more detailed and include the possible impacts of the change. In addition, the demand deadline is usually not feasible with the development "effort", which is something that is discrepant when it comes to deliveries

Person 11: It would be interesting if the PO had more choice in the activities to be prioritized per sprint, with the Tech Lead being in charge of defining the best developer. Perhaps this is the case and I haven't noticed.

## Survey 2 - 17/03 to 28/03

1. How do work demands come to you on a daily basis?

(Describe the process, including where you receive it and how this assignment happens).

Person 1: They come to me through my leader who has already evaluated the specification and arrived at the point where it is feasible.

Person 2: I receive demands from my Tech Lead in calls or dailys, which before reaching my tech lead are raised by the clients and specified by the PO. But there have been cases where I've listened to the pain directly from the client and managed to map out the solution and develop it.

Person 3: Through my line manager or the robot that sends the issues

Person 4: The demands come through the PO/Tech lead. This happens more than once a week. The PO specifies the projects and aligns them with the Tech Lead who distributes them among the available devs.

Person 5: They arrive via specification reports, these reports are included in our Confluence and then shared with the developers assigned to the project, meetings are usually held to go over the specification with the devs.

Person 6: Through Confluence, I receive how the flow is and how I should set up the solution. Then I can assemble the cards according to what I want to develop.

Person 7: The demands arrive via the PO and Tech Lead and the Developers create the tasks.

Person 8: Demands usually arrive after a conversation with clients or collaborators, and are then separated by priority for each team member.

Person 9: Through management tools with agile dashboards (Scrum and Kanbam)

Person 10: Via team chat or allocation from the superior via new project specifications

Person 11: Through tech leads and requirements analysts

Person 12: Card assignment in the Jira system. In the Daily Meetings we have, we talk about completed demands, future demands, cards are created, assigned to the managers of each team (techleads), and 'child' cards are created to assign to whoever picks up the task.

Person 13: Receipt occurs through Jira, most of the time I identify the tasks that need to be done and assign myself to them by notifying the tech lead. There are also tasks that are defined with the team, tasks aimed at improving workflow, improving the company's projects or activities aimed at improving the delivery of demands.

Person 14: I usually look in Jira and check with the Tech Lead about the possibility of starting the task

Person 15: I ask my Tech Lead if there's a task I can take on, or I look at 5 Jira boards, one for each client, and ask them if I can take on the task that's available.

2. Do you think that the way demands come to you currently works well or could it be improved?
- It works well, I don't see any need for change.
- It works, but it could be better.
- It doesn't work well, it needs improvement.
- Others...
-

Person 1: It works well, I don't see any need for change.

Person 2: It works, but it could be better.

Person 3: It works, but it could be better.

Person 4: It works, but it could be better.

Person 5: It works well, I don't see any need for changes.

Person 6: It works, but it could be better.

Person 7: It works, but it could be better.

Person 8: It works, but it could be better.

Person 9: It works, but it could be better.

Person 10: It works, but it could be better.

Person 11: It works, but it could be better.

Person 12: It works well, I don't see any need for change.

Person 13: It works well, I don't see any need for change.

Person 14: It works, but it could be better.

Person 15: It works, but it could be better.

3. If you think it could be improved, how could this process be made more efficient?

Person 1: No answer

Person 2: I believe that the specification should be read by the tech lead with a more critical eye. Sometimes my fellow developers and I have had to figure out where to look for certain information, such as databases and so on, which ends up hindering the delivery of the project.

Person 3: Tasks could arrive by robo, currently they arrive by e-mail

Person 4: I think the tasks could be divided according not only to the availability of the devs, but also to their level of experience.
It would be nice to have greater visibility of quick wins, so that any time gap in the sprint could be converted into a task, without idle time between one project and another or large projects being blocked.

Person 5: No answer

Person 6: the cards could be ready-made, with a complete step-by-step description of what needs to be done. For example, test the integration with table x in the lake, match the data in the table with file y. This way, I could just worry about putting what was requested into the logic of the code. That way, I could just worry about putting what was requested into the logic of the code. It would also be interesting to have homologation cards to make testing easier and to deliver the card in develop done closer to what it should be.

Person 7: I think it would be better for POs to create tasks for each User Story. That way, the project would be better monitored.

Person 8: Understand all the points before starting to pass on demands!

Person 9: Documentation showing what is being done and how it is being done in the code in line with the business specification documentation. This would optimize the time that is wasted on any checks that need to be made.

Person 10: Specifications with greater depth of flow and the participation of a developer in the discovery process. Improve project scope limitation and better filter out projects that make sense.

Person 11: No answer

Person 12: At the moment I think this treadmill is quite functional.

Person 13: No answer

Person 14: sprint start and duration notifications could be shown in a clearer and more accessible way through a spreadsheet, table or any other tool that would help to clearly inform the time of action, especially in scenarios where several sprints occur in parallel. The process of notifying demands is still very “manual”, there could be a platform with all the tasks containing an option for the dev to “signal” that they want to start that activity and then the person responsible (in my case the Tech Lead) would be notified with the option of approving or rejecting that request.

Person 15: If someone inserts a new card, inform them that there is a new task, instead of me looking board by board to see if there is a new card, there are days when there is no new card all day.

4. How often do you receive work demands?
- Daily
- Weekly
- Monthly
- Other...

Person 1: Other: I receive demands like a treadmill, where a new project arrives only when I've finished the current one.

Person 2: Other: Every two weeks

Person 3: Daily

Person 4: Daily

Person 5: Weekly

Person 6: Monthly

Person 7: Daily

Person 8: Daily

Person 9: Daily

Person 10: Daily

Person 11: Weekly

Person 12: Weekly

Person 13: Other: It depends with the number of tasks and sprint, the sprint is over and requirements are still being raised and cards are being created I am free from Jira board tasks, but there are still tasks defined with the team that need to be touched.

Person 14: Other: almost daily. I start an activity after finishing another, which takes an average of 1 to 4 days depending on its complexity.

Person 15: Daily

5. Do you feel you have the autonomy to organize your own demands or are you totally dependent on others?

- I have complete autonomy
- I have some autonomy, but I follow a fixed direction
- I am totally dependent on others

Person 1: I have total autonomy

Person 2: I have some autonomy, but I follow a fixed direction

Person 3: I have some autonomy, but follow a fixed direction

Person 4: I have some autonomy, but follow a fixed direction

Person 5: I have complete autonomy

Person 6: I have complete autonomy

Person 7: I have total autonomy

Person 8: I have total autonomy

Person 9: I have some autonomy, but I follow a fixed direction

Person 10: I have total autonomy

Person 11: I have some autonomy, but I follow a fixed direction

Person 12: I have some autonomy, but I follow a fixed direction

Person 13: I have some autonomy, but I follow a fixed direction

Person 14: I have some autonomy, but I follow a fixed direction

Person 15: I am totally dependent on third parties