# SCRUM HAS EVOLVED


**3rd Edition**

# Acknowledgements

I am grateful to my friends for listening to my “fiery speeches” about agility and for encouraging me to write this book.

I would like to thank the companies Brasfond, Aon, Fidelity, SQLive, Ultralub Química, Acmos, Tivit, Protege, and Credsystem for giving me the opportunity to work with agile methodologies and frameworks at different times in my career, and to uncle Mark Zuckerberg and his AI
for the images (I erase the Meta logo because I don't want to give the false impression that it endorses my articles) and for the spelling correction of several texts.

I dedicate this booklet to my son, Gui, who with his perspective as a person with autism is sometimes able to see what neurotypical people do not see.

**MARCELLO DIAS**
**PSM1,PSPO1,PSK1 e SPS**

# Introduction

This book is nothing more than a collection of articles I wrote on LinkedIn.

Some topics are deliberately covered superficially, such as graphs and tools, so that the book does not become too long and boring. The reader can use Google to delve deeper if they wish.

I used to be a fierce critic of Scrum, as I thought it was rigid compared to Lean Kanban (I wasn't the one who changed it, it was the Scrum guide).

Scrum has evolved to , but many Scrum Masters don't, yet propagate outdated thinking, myths and misinterpretations.

Ron Jeffries, inventor of story points, no longer uses them.

If the Review served to deliver something, it would be called a delivery meeting. Scrum works with continuous delivery, the Review serves to review what was done in a period of time encapsulated by the Sprint.

Stakeholders are not contagious, they do not need to be removed of the Scrum team.

QA is just a developer specialized in testing, not a babysitter.

Scrum “didn’t work” for you? Get updated.

The bike is not to blame for the fall, it is the cyclist. Scrum is a framework that has been tested and approved in a wide range of situations.

# INDEX

# What makes a good Product Backlog?

***This is perhaps the most important chapter of this book.***

The Product Backlog, or list of product features and requirements, is the main element of Scrum. It works as a list that is always changing and organized according to priorities, containing everything that may be necessary for the development of the product.

A good backlog is not just a to-do list, but a tool
strategic that guides the continuous progress of the product, ensuring that the team delivers value incrementally.

Let's take a look at the main components of a well-designed backlog.

Let's talk about its structure, the criteria to guarantee its quality and the best practices to constantly improve it.

## 1. Components of a Good Product Backlog

According to **Scrum.org,** an effective backlog should contain the following elements:

### a) Clear and Well-Defined Items

Each backlog item (usually represented as **User Stories, Epics or Tasks)** should be:

**-Valuable:** Deliver tangible benefit to the user or business.

**-Independent:** Minimize dependencies with other items.

**-Negotiable:** Open to discussion and adaptation.

**-Estimatable:** Allow the team to assess its size and complexity.

**-Small:** Broken down into manageable chunks to facilitate delivery in sprints.

**-Testable:** Have clear acceptance criteria for validation.

## b) Value-Based Prioritization

The **Product Owner (PO)** is responsible for prioritizing items based on in:

**-Value for the customer/business** (ROI - Return on Investment)

**-Risks and dependencies.**

**-Needs , market trends and feedback from stakeholders.**

**-Alignment with the product vision.**

We will have a chapter just on Backlog prioritization, where we will present several standard market techniques.

## c) Defined Acceptance Criteria

Each item must have **clear acceptance criteria** that define when a feature is "done" (Definition of Done - DoD). Example:

Below is an example of acceptance criteria for the "Add to Cart" functionality.

-The system displays a visual confirmation (e.g. popup or updated icon).

-The cart displays the updated total price.

-Removed items are deleted immediately.

-Persists items if the user logs out and back in (with active session).

Below is an excellent article on acceptance criteria:

https://www.atlassian.com/work-management/project-management/acceptance-criteria

## d) Continuous Refinement

The backlog is not static, it evolves with:

-New discoveries (user feedback, market changes).

-Decomposition of large items.

-Removal of obsolete items.

Regular refinement sessions with the team are recommended to keep the backlog relevant.

## 2. Best Practices for Maintaining an Efficient Backlog

### a) Transparency and Accessibility

-Everyone must understand the backlog.

-Tools like Jira, Trello or Azure DevOps help with visualization.

b) Balancing Detail and Flexibility

-Items at the top (next sprints) should be more detailed.

-Items at the end of the backlog should be more abstract, aiming to maintain flexibility.

### c) Collaboration between PO and Dev Team

-The PO defines the "what" (value), the team suggests the "how".

Frequent discussions avoid misunderstandings.

d) Constant Review and Adaptation

-The backlog should reflect learnings from each Sprint Review.

-Changes in the market or technology may require adjustments.

## 3. Common mistakes and how to avoid them

| Problem | Solution |
| --- | --- |
| Very large and disorganized backlog | Refine and decompose items regularly |
| Inconsistent prioritization | Define clear value criteria (e.g. ROI, risk) |
| Lack of clarity in items | Ensure well-written user stories and acceptance criteria |
| Little collaboration with the team | Involve devs in backlog refinement |

# 4. Metaphors about the product backlog

During my years of involvement with Scrum I have come across several analogies regarding the backlog, most of them didn't leave my head, some I found in books, others in forums.


## a) The Backlog as a Compass

Some Agile experts compare the backlog to a compass:

-Clear direction: Just as a compass points north, the backlog indicates what needs to be done to achieve the product vision.

-Continuous adjustment: If the wind changes (new demands), the PO readjusts the route, but the final objective remains.

A well-maintained and prioritized backlog is like a compass for the team Scrum – it keeps everyone sailing in the same direction, regardless of adverse conditions.

### b) The Backlog as a Beacon

-Lights the way: Shows the next steps, preventing the team from getting lost in unnecessary features.

-Warns about hazards: High-risk items or critical dependencies are like rocks in the sea.

### c) The Backlog as a Garden

-It needs constant care: If it is not refined, it becomes a "thickness" of forgotten items.

### d) The Backlog as a toolbox

- Materials (tasks) are stored and wait their turn to be used (developed) to build the final product.

## Conclusion

The Backlog is dynamic, transparent and value-oriented. It must be refined, prioritized and aligned with the product vision, ensuring useful increments with each sprint.

The key to an effective backlog is collaboration between Product Owner, developers and stakeholders, focused on the value delivered to the end user.

# Hard and soft skills needed to be a good Scrum Master.

The Scrum Master role is crucial to the success of agile groups. More than Rather than a facilitator, he acts as a servant leader, removing impediments, promoting continuous improvement, and ensuring that Scrum is well understood and applied.

Both Scrum.org and the books "Scrum Mastery" (written by Geoff Watts) and "Coaching Agile Teams" (written by Lyssa Adkins) emphasize the need for a balance of technical skills and interpersonal skills for an effective Scrum Master.

## 1. Hard Skills needed for a Scrum Master

Hard skills are related to the technical skills and specialized knowledge that a Scrum Master must acquire and master:

**a) Mastery of the Scrum Framework**

-Deep understanding of **Scrum events** (Sprint, Daily, Review, Retrospective, Planning).

-Knowledge of **Scrum artifacts** (Product Backlog, Sprint Backlog, Increment).

-Understanding of **roles and responsibilities** (Scrum Master, Product Owner, Developers).

**b) Agile Metrics and Tools**

-Ability to track metrics such as **Cycle Time, Throughput, Lead time**

-Familiarity with tools like **Jira, Trello, Azure DevOps.**

-Knowledge of **backlog refinement** techniques (User Stories, Estimation Techniques).

**c) Knowledge of Other Agile and Lean Methods**

-Notions of **Kanban, XP (Extreme Programming), Lean.**

-Understanding of **workflows and continuous improvement (Kaizen).**

**d) Resolution of Technical Problems and Impediments**

-Ability to identify and remove **technical and organizational impediments.**

-Ability to assist the team in **resolving prioritization conflicts.**


## 2. Fundamental soft skills for a Scrum Master.

Interpersonal skills, known as soft skills, are essential for the Scrum Master, as they enable leadership without the need for formal authority. These skills are essential to inspire the team and foster an environment of effective collaboration.

First of all, a Scrum Master must know how to recognize his or her mistakes, be humble enough to say, “I don’t know, I’ll look into it,” and be able to apologize. One example is worth more than a thousand lectures and five hundred hours of coaching.

**a) Servant Leadership**

-Focus on **serving the team,** removing obstacles and facilitating their growth.

**-Empower** team members to be self-manageable.

**b) Effective Communication**

-Know how **to listen actively** and transmit information clearly.

-Facilitate productive discussions in **Scrum ceremonies.**

-Adapt communication to different stakeholders (team, PO, leadership).

.

**c) Conflict Facilitation and Mediation**

-Conduct meetings in a **neutral and productive manner.**

-Mediate conflicts within the team, promoting a **safe and collaborative.**

**d) Empathy and Emotional Intelligence**

-Understand the **needs and frustrations** of the team.

-Help develop **healthy relationships** within the team.

**e) Coaching and Mentoring**

-Apply **agile coaching** techniques to help the team evolve.

**-Mentor** the Product Owner and the organization in the adoption of agile practices.

**f) Adaptability and Resilience**

-Deal with changes and resistance proactively .

-Promote a **growth mindset .**

## 3. How to Develop These Skills?

**-Certifications:** Take courses such as **PSM (Scrum.org) or CSM (Scrum Alliance).**

**-Continuous Practice:** Apply Scrum to real projects and seek feedback.

**-Reading and Study:** Delve into books like **"Scrum Mastery"-** Geoff Watts e **"Coaching Agile Teams"-Lyssa** Adkins.

**-Networking:** Participate in agile communities.

To finish, I suggest reading the article: https://
www.scrum.org/resources/blog/28-characteristics-great-scrum-master

***Conclusion: Being*** an effective Scrum Master requires more than just having a basic understanding of the Scrum framework. It is essential find a balance between hard skills, which represent technical competencies, and soft skills, which encompass skills such as leadership, communication and empathy. As Geoff Watts and Lyssa Adkins point out, the authentic Scrum Master is the professional who motivates the team, eliminates barriers and fosters an environment conducive to constant learning. Invest in improving these skills and become a true agent of change within your organization!

# The “guesser” Scrum Master.

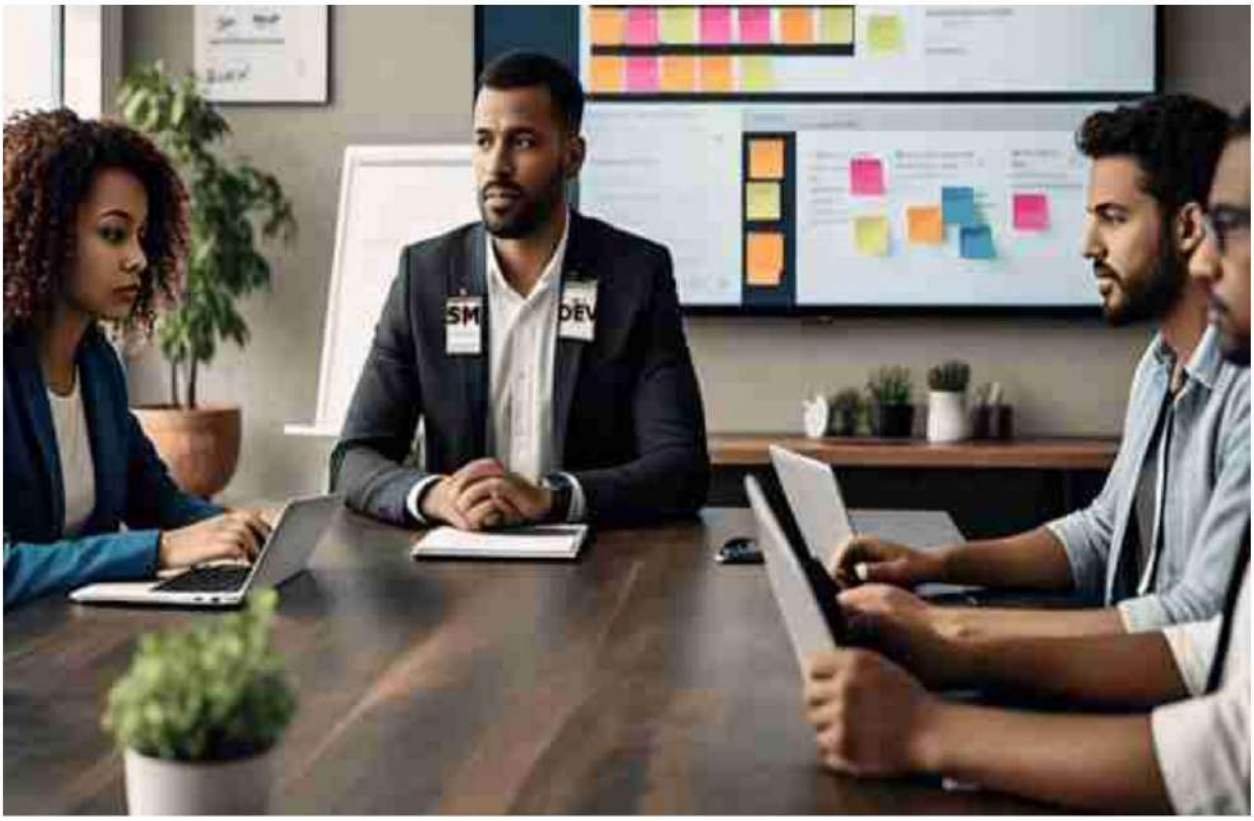
I decided to write this article because I participated in a Scrum team, whose Scrum Master had been a developer for a very short time, and apparently, he liked the role of developer more than that of Scrum Master.

Sometimes his opinions were even interesting, but this generated a certain animosity, especially with a less experienced developer, who usually felt offended.

It is worth mentioning that this Scrum Master was an “odd one out” and had not worked as a developer in the company.

The role of the Scrum Master is critical to the effectiveness of the Scrum team.

Their responsibilities are to remove impediments, facilitate ceremonies, and ensure that Scrum principles are well understood and practiced. However, when the Scrum Master has a background in development, the question arises: should he contribute technical opinions or abstain so as not to compromise the team's autonomy?



### The Scrum master must be a servant leader, whose main focus is:

-Help everyone understand and adopt Scrum correctly.

-Act as a facilitator for the Scrum team and the organization.

-Remove obstacles that hinder the team's progress.

-Promote self-organization and empirical thinking.

According to the Scrum Guide 2020: The Scrum Master is not a traditional project manager or technical leader, but rather someone who helps the team become more effective. This means not completely avoiding critical discussions, but also not imposing his technical vision.

## Scrum Master with Technical Background: Benefits and Risks

Scrum Masters with previous experience as developers can bring significant value — but also challenges. Here are the key takeaways:

### Benefits

-In-depth Technical Understanding: facilitates the identification of real impediments, especially those involving architecture, technical debt or continuous integration.

-Credibility with the Team: Practical experience increases trust and respect, making it easier to facilitate complex debates.

-More Efficient Facilitation: By understanding technical language, you can connect the right people, articulate better questions and promote collaborative solutions.

### Risks

-Micromanagement: can start to direct decisions or suggest ready-made solutions, suffocating the team's autonomy.

-Technical Bias: there is a risk of favoring technologies or approaches based on personal experiences, without considering the current context.

-Role Confusion: the team may confuse the Scrum Master with an informal Tech Lead, leading to the centralization of decisions.

Example: In a sprint planning, the Scrum Master repeatedly intervenes to say how a feature should be implemented. Result: the team stops proposing alternatives, and depending on the case, discussions may even occur.

*When should a Scrum Master (or not) give an opinion, outside of his/her responsibilities?*

The Scrum Master can provide suggestions and guidance, but the most important thing is to always pay attention to the process, not the technical content. Therefore, it is essential to know the right time to make these contributions.

### Situations in which giving an opinion is appropriate:

-When explicitly requested by the team.

-To warn of possible risks or technical impediments (e.g.: “This solution may hinder continuous deployment?”).

-In discussions about agile engineering practices, such as Definition of Done, continuous integration, or code quality.

-To trigger reflections, not to dictate solutions (e.g.: “Have you considered how this impacts system performance?”).

## Situations where it is better for him to keep quiet:

-In technical decisions that are exclusively the responsibility of the development team.

-When the Scrum Master interrupts or dominates discussions, limiting exchange between developers.

-When there is no direct invitation to give an opinion, especially if the team is already collaborating effectively.

## Using psychology:

Instead of directly suggesting a technology, ask questions that lead the team to think: *“What would be the impact of using this approach on security?”*

A technical Scrum Master does not need to sit back and be quiet, but must use his or her influence wisely. His or her knowledge should be used to provide light guidance, intelligent facilitation, and strategic questioning, always in a collaborative manner, never as an imposition.

> Mike Cohn, in *Succeeding with Agile,* highlights that the Scrum Master should be “a guide on the journey of continuous improvement”, not the decision-maker.

This balance is what makes a simple hunch become a truly important contribution, preserving the team's autonomy and strengthening self-organization — one of the pillars of Scrum.

It is important to highlight that a Scrum Master can simultaneously perform the role of developer, as long as this condition is well defined.

## Conclusion

Yes, opinions are welcome—but in moderation, with purpose, and with active listening. A thoughtful Scrum Master who knows when to speak up and, most importantly, when to facilitate others’ expression can be an excellent ally. Technical expertise is a competitive advantage, as long as it is used in the service of the team, not as a shortcut to control. The Scrum Master, acting as a mentor and guide rather than a technical decision-maker, can be a catalyst for building stronger, more autonomous, and productive teams.

# In Scrum we work with continuous delivery.

I was previously a vocal critic of Scrum.
However, it wasn't me who changed, it was Scrum, or at least everyone's perception of Scrum.
I have always advocated constant improvement, inspection and adaptation.

I have always opted for prototyping over Waterfall, long before the agile manifesto existed.

Prototyping is a kind of Scrum without roles, responsibilities, pillars, values and ceremonies, which depended a lot on the talent and common sense of each person, but was already superior to waterfall.

Most companies mistakenly associate increment delivery with
at the end of the Sprint, as if the Review were an altar to the God Agilitos.

If Agilitos does not obtain his sacrifice, the entire tribe will be struck by his wrath.

Scrum has never been very popular in Asia, where teams have always opted for Kanban, with its Lean principles.
However, now that Scrum has incorporated Kanban techniques, and further reinforced Lean principles, it's like having the best of both worlds.
worlds, I went from critic to admirer.

## WORKING WITH CONTINUOUS DELIVERY IN SCRUM.

Most current Scrum teams still associate delivery with the altar of Agilitos (Review), making many feel tempted to commit to the minimum of "story points".

Nowadays, some teams need an army of Oompa-Loompas to implement a simple registration system. Even Waterfall would be superior to that.

When combining Kanban with Scrum, estimating is not the most important thing, you break down the story, when you are "almost certain" that it will be feasible to finish it in a Sprint (meet the SLE), you limit the amount of PBIÿ in the production line (WIP), and the focus shifts to continuous delivery.

It can be sent to production before or after the review. Anything that is not ready returns to the product backlog, usually being resumed in the next Sprint.

But it is worth highlighting that selecting the content of the Sprint Backlog through the analysis of Throughput and Cycle Time, establishing a WIP,

the probability of delivering everything selected in Planning increases greatly.

It is worth remembering that most agile methodologies and frameworks date back to the 90s, where an increment almost always equaled a monolith, today with the advent of microservices, often the team Scrum works in several increments in a Sprint, it makes no sense to wait for the Review to deliver everything, when often the increment generates value, and is eagerly awaited in production.

In a scenario like this, it is even more necessary to control dependencies, remembering that there are graphs that show the dependency between sprint backlog items.


## Conclusion

If the team finished a story early, and the delivery generates value, why wait for review? If the PO so decides, the increment can go to production before review.

By ceasing to use metrics with a high level of imprecision, such as story points, and starting to focus on throughput, cycle time and establishing a work in progress (WIP) limit, the chance of completing all tasks during the Sprint increases considerably.
Much more confidence in assembling the Sprint Backlog!!!

What was not presented in the review on Friday may very well be ready on the following Monday (next sprint) and go into production. No one died, so the team stops committing to little, and transparency, one of the pillars of Scrum, is reinforced.

# HR as an enabler of agile culture.

When companies adopt Scrum, they are not only changing the way they work, but also undergoing a true cultural transformation. This means that teams need to be self-managed, collaborative and highly efficient. In this scenario, the role of Human Resources (HR) becomes even more important, as it is responsible for ensuring that the professionals hired have not only the necessary technical skills, but also the values and interpersonal skills (soft skills) essential to succeed in an agile environment.

## High turnover: a consequence of poorly aligned hiring

One of the biggest problems in companies that implement Scrum is high employee turnover, often caused by flaws in the recruitment process. HR often receives vague or poorly defined demands from managers, resulting in misguided hiring.

"A poorly hired employee is a future fired employee" – If the person
does not fit into the team's culture, values or expectations, his departure is only a matter of time.

Lack of clarity in vacancies – Companies often look for a “generalist who knows a little about everything”, when what they really need is a specialist with in-depth knowledge in key areas, everyone can learn everything, but quality comes with experience, the Scrum team must have the necessary knowledge to

perform your tasks accurately.

The following article discusses balancing generalist and specialist knowledge on a Scrum team: Balancing Generalists and Specialists– Building Successful Agile Teams - InfoQ .

AI assistants today minimize the need to worry about generalists, quality is non-negotiable, hire the specialists you need, generalism (which will rarely be used) is easy

to be achieved.

In Scrum, efficient teams are formed by professionals who master their disciplines and collaborate in a synchronized manner. A developer who has superficial knowledge of several languages but does not solve

complex problems in any of them, can become a bottleneck in
instead of an asset.

**The Danger of the "Technically Wonderful" with Bad Soft Skills.**

A common mistake in selection processes is to overvalue technical skills and
ignore cultural fit and soft skills. In Scrum, trust and respect are fundamental –
a professional who is extremely technically competent but toxic in relationships can
destroy the team dynamic.

**Corporate Serial Killers** – There are professionals
who are brilliant individually but disastrous as a team: arrogant, inflexible or
disrespectful. In Scrum, this is unacceptable.

**Soft skills are not optional** – Communication, empathy, collaboration and
resilience are just as important as technical skills.

**Psychometric tests and cultural fit should be a priority**
– A candidate may be a programming genius, but if he or she doesn't align with
the company culture, his or her performance will be limited.

**Empowerment in Scrum Requires Strategic HR.**

For Scrum to work, HR must go beyond traditional recruiting and act as an enabler of agile culture:

**Define profiles clearly** – Work together with Product Owners and Scrum Masters to identify the technical and behavioral skills required.

**Prioritize cultural fit and soft skills** – Implement dynamics, behavioral interviews and psychometric assessments to ensure that the professional aligns with the team's values.

**Avoid the generalist trap** – Seek out specialists who bring technical depth rather than superficiality across multiple areas.

**Promote an environment of trust** – Ensure that employees have autonomy, but also responsibility and mutual respect.

**Conclusion: The** implementation of Scrum is only successful when HR assumes a strategic role, going beyond simple hiring and ensuring that professionals have the appropriate technique and temperament.

Bad hires lead to future layoffs – And in Scrum, where collaboration is essential, a recruiting mistake can be costly.

Technical skills are necessary but not sufficient – A toxic "genius" can break team trust, while a technically sound and collaborative professional elevates the entire team.

Therefore, HR is not just a bureaucratic support – it is the foundation that sustains high-performance agile teams. When recruiting is done
With criteria, clarity and attention to cultural fit, Scrum is strengthened and the company reaps , and the rewards of a productive, innovative and harmonious environment.

# Scrum: It is necessary to change the company culture.

In Scrum, competent, self-organized, multidisciplinary, inspected and empowered teams are formed.
The philosophy is that many are better than one.
There is hierarchy, but it is based on respect and purpose.

If your organization’s culture is based on rigid oversight structures, a lot of detailed reporting and planning, and if the priority is bureaucracy rather than the product, then implementing agile methods or frameworks can be a big challenge. Scrum is not a replacement for HR; it’s important to hire professionals you can trust.

As Steve Jobs said, "There's no point in hiring smart people and telling them what to do; we hire smart people to tell us what to do!"

In companies that have truly adopted Scrum, in the correct way, with employees well selected by HR, with Scrum Masters certified by a serious institution such as Scrum.org, for example, what is noticeable is that the members of the Scrum team embody the spirit of

owner, one becomes a supporter of the other's improvement, they become true guard dogs of the company's values, they feel robbed when someone slacks off, as the sense of belonging grows within them, as they become proud of the product they develop.

Daily communication and continuous peer-to-peer inspection promote a relentless pursuit of comprehensive understanding of the product under your responsibility, as well as contributing to professional development.
When knowledge and technical information are shared, everyone improves, increasing their value as professionals, while the company does not become dependent on a single person.

Consider the previous illustration in contrast to a model where only the manager is in charge of defining quality guidelines, establishing deadlines and determine technical solutions.

I didn't believe it, the experience with Scrum changed my thinking.

I've seen developers who were "pushing their careers with abandon" become bookworms.

We must respect the traumas that each company carries, the word empowerment is the one that causes the most fear, when the focus should be on inspection, adaptation, transparency and quality.

Many business owners have been robbed, hired bad professionals, and thrown money away, especially in Brazil. Many times, it is necessary to implement Scrum right after the company has fallen into some “PicareTIagem” (rogue companies existing in the IT area), that is, with a high level of distrust.

**Conclusion:** Scrum increases everyone's sense of responsibility for the product they develop and encourages continuous improvement.

# How should the Scrum Master act in a tense retrospective?

## The Retrô is the ideal ceremony for the Scrum team to analyze what went well, what was not successful and how it can evolve.

However, if the atmosphere is tense—particularly when a
developer is the target of criticism from the group,
The Scrum Master needs to act wisely to convert discord
in learning and ensure a safe and cooperative environment.
In two months, perhaps, no one will remember what the argument was about, but
If the level of discussion drops too low, the consequences could be
lasting.

### The Scrum Master's Role in a Conflictual Retrospective

The Scrum Master is the guardian of the process and must ensure that the
Retrospectives should be productive, respectful and focused on improvements, not
blame. If the team is directing criticism in a hostile way at a single
member, the Scrum Master must intervene strategically.

## 1. Establish a safe environment

Remember the Scrum values (respect, openness, courage): Before
to begin with, reinforce that the goal is to improve as a team, not
point fingers.

Use facilitation techniques: Dynamics such as "Start, Stop, Continue" or "Sailboat Retrospective" help keep the focus on actions, not on people.

Establish clear rules: Set expectations for dialogue, such as mutual consideration and concentration on solutions.

## 2. Redirect the discussion to facts, not judgments

Avoiding generalizations: If someone says "You're always late the progress of the Sprint", the Scrum Master should rephrase the question to "What is the reason for this delay? How can we do it next time?" time, so that this delay does not happen again?"

Focus on the impact, not the person: Ask the question "What way did this influence the team?" instead of "Why did you take this action?"

Employ the "root cause analysis" approach: Encourage the team to identify the fundamental reasons for the problem.

## 3. Ensure everyone has a voice

Give space to the person being criticized: Allow him/her to explain your point of view without interruptions.

Avoid "group attack": If several members are criticizing a colleague, the Scrum Master can intervene with questions such as:

"What could each of us have done differently?"

"How can we support this member to avoid the problem in "in the future?"

Use the "constructive feedback" technique: Encourage team members to team to give specific, behavioral and improvement-focused feedback.

## 4. Turn conflict into improvement

Ask "What did we learn from this?": Encourage the team to extract lessons, not just complaining.

Define concrete actions: If the problem was a technical error, propose peers for code review or training.

Create an action plan: Define concrete steps to avoid similar problems in the future.

### 5. End by focusing on the future

Reaffirm the team's commitment: Remember that the strength of the group comes of all, that everyone must row in the same direction.

Schedule follow-ups: If necessary, schedule a conversation one-on-one with the developer for support.

Celebrate successes: Recognize and celebrate team successes, even if small.

## Conclusion

A tense retro can turn into a significant chance of development if the Scrum Master acts as a facilitator, preserving respect and guiding the team in search of solutions. The priority should be to improve processes, rather than pointing out fingers.

If the environment remains tense, an individual dialogue with the participants may be necessary to reaffirm the Scrum values.

# The fake agile, or Scrum with reins.

The race for agile transformation in Brazilian companies has a frightening truth: much of it is pure theater. Companies adopt Kanban boards and Scrum ceremonies not out of methodological conviction,
but due to market pressure. The result? The Scrum Master becomes a "mini-project manager", rigid schedules are disguised as "sprints"
flexible", and the Waterfall culture remains untouched.

The "Scrum Master" (now in the position of manager) requests activities and makes updates in spreadsheets, while meetings become personal evaluation rituals without effective changes, since the vast majority of decisions still require authorization from higher levels.

## Scrum Master as a "Little Manager and Master of Ceremonies":

A true Scrum Master acts as a facilitator and obstacle eliminator. However, in practice, many companies turn him into a
a deadline watcher, focusing more on meeting strict schedules than on spreading agile culture.

Instead of training the corporation on the values and pillars of Scrum, he becomes a spokesperson for the Hierarchy: He passes on demands from superiors instead of making the information and the focus on the customer circulate in both directions: Top-down and bottom-up.

## The table below highlights the contrast: Authentic Function vs. Distortion

| Authentic Function | Distortion |
|---|---|
| Remove impediments | Charge deliveries |
| Facilitates team decisions | Relays management orders |
| Promotes self-organization | Controls tasks individually |

The masked Waterfall mindset and the obsession with absolute predictability still dominate many companies.
Many successful entrepreneurs (who sometimes started alone, or with little collaboration, some using prototyping and not waterfall) follow the line of thought: “I got here without this @#!$ of agile, empowerment is the whole thing!”, but they don’t see that:

1-It is not so easy to make an employee incorporate the spirit of an owner.
2-It is very difficult to maintain quality and keep an eye on the customer when business grows.

Steve Blank, creator of the term "Innovation Theater", warns: "Companies undertake initiatives that appear to be innovation, but create little real value".

The consequences for the professionals involved include fear, dismissals and a feeling of guilt that is often unfair.
Many Scrum Masters end up accepting this situation out of fear of losing their jobs. In order to keep their positions, they become "little managers", which is not their original role.

**Conclusion:** Agile is Culture, not Cosmetics, companies that pretend to be Agile practices pay a high price: wasted resources, loss of competitiveness, and teams with high turnover. The path to change requires cultural awareness and managerial courage.
While organizations seek to "look modern," those that truly embrace agility reap the rewards: market-aligned products, motivated teams, and sustainable innovation.
The choice is clear: be agile, or pretend to be and succumb to harsh reality.

# Page left intentionally blank

## The Scrum Master must be the agent promoting transformation, or he may become a laughing stock.

### The day-to-day life of a Scrum master: Challenges and lessons learned

A Scrum master is a key professional in software development teams that adopt the Scrum agile framework. Their role is to ensure that the team follows Scrum principles and practices, removing obstacles and continuously improving the process. However, this task is not simple and involves a series of challenges, especially when it comes to implementing Changes.

## Implementing changes while respecting the culture of the company

An effective Scrum Master must be able to implement changes gradually and respectfully, taking into account the culture of each company. It is essential to understand that each organization has its own history, values, and way of working, and ignoring these aspects can lead to resistance and even failure.

If a Scrum Master tries to implement changes too quickly without considering the company culture, he or she may face strong resistance from employees and even be fired. As Ken Schwaber, co-creator of Scrum, once said, “A dead Scrum Master is a useless Scrum Master.” This may seem harsh, but it reflects the reality that a Scrum Master must be able to navigate the political and cultural complexities of an organization in order to be successful.

Now, simply giving in to the situation will lead, in the long run, to dismissal, as by becoming part of the landscape the Scrum Master will be seen as useless.

Scrum is an agile framework that adapts to any size and type of company. It can be implemented in small startups or large corporations, and its framework is flexible enough to be adapted to
specific needs of each team. Furthermore, in smaller teams or
In the early stages of Scrum adoption, it is common for a developer to take on the role of Scrum Master.

## The challenge of being a Scrum Master

Many Scrum Masters end up succumbing to the pressure of implementing “change with a leash” which can lead to a situation where they become useless or mere facilitators of Scrum ceremonies. Instead
to act as agents of change, they may limit themselves to holding meetings and following bureaucratic procedures.

This is because, in an attempt to keep their jobs at all costs, some Scrum Masters end up losing sight of their primary role: to continually improve the process and ensure the team’s adherence to the Scrum principles. They can become so focused on maintaining stability and avoiding conflict that they end up not fulfilling their role effectively.

The Role of the Scrum Master

A Scrum Master must be a servant leader, able to inspire and motivate
the team to work collaboratively and focused on delivering value to the customer. They must be able to identify obstacles and work towards them.

remove them, improve communication within the team and ensure that everyone are aligned with the project objectives.
Additionally, a Scrum Master must be a facilitator of continuous learning, helping the team reflect on its own process and identify opportunities for improvement. This requires a combination of technical, communication, and leadership skills.

## The Scrum Master cannot conform to old impediments.

The Scrum Master cannot conform to impediments **“written in the rock”,** he must know how to show cost and benefit with explanations as:

-“You knew that if we had a homologation environment more similar to the production environment, developers would be more productive, as they would not have to generate a lot of data, and we would also reduce the possibility of Rollbacks in the production environment.”

— “The development tool, such and such, is compatible with the latest version of the DBMS, such and such.”

If the Scrum Master behaves like a “scaredy-cat” who accepts impediments as natural things in life, he/she will become a laughing stock in the Scrum team and will succumb over time.

## Conclusion

Being a Scrum Master is a complex challenge that requires specific skills and competencies. Implementing changes in a respectful and gradual manner, considering the company's culture, is essential for the success.

However, many Scrum Masters end up succumbing to the pressure and becoming useless or mere facilitators of ceremonies. It is essential that Scrum Masters maintain their focus on continually improving the process and ensuring team adherence to Scrum principles.

Scrum is a flexible agile framework that can be adapted to any type of company, and a developer can take on the role of Scrum Master in smaller teams.

# With weak PO there is no team good.

There is a very crude saying that I will not repeat here, about people who choose their partners taking into account only physical attributes, disregarding character and intelligence, and then complain about marriage for the rest of their lives.

A very common mistake is to choose the PO expecting that he will learn from the stakeholders and the scrum master.

The “Anyone can be a PO” mindset has led to non-value-added products and endless hours of rework.

The PO has to be “the guy”, has to understand a lot about the target area of the product, has to be respected by stakeholders for his knowledge, and not be a simple office boy for them.

The PO is a value maximizer, product owners should get Scrum certification, especially those who have been managers.

Managers often become terrible product owners. They need to go through a detox session and understand that respect comes from knowledge, dialogue, and the group's sense of success, not from the "card-grabbing" effect.

**Conclusion: A** PO who does not master the product area and does not Failing to identify opportunities often results in a lot of rework and frustration.

“Fail fast” doesn’t mean throwing your company’s time and money away. It’s adaptable to any size company.

# How to select and train Product owners?

The Product Owner, known as PO, plays an essential role in Scrum. He acts as the liaison between those involved in the project, stakeholders, and the development team. The main task of the PO is to ensure that the final product offers maximum value, keeping the backlog properly structured and aligned with business demands.

However, selecting and training a good PO is not a simple process. Many organizations face challenges in choosing the right person or in preparing them properly for this position. In this article, we will discuss how to make this selection and provide guidance for train your Product Owner. We will also explain why the Professional Scrum Product Owner, or PSPO, certification can be an important differentiator in this process.

# 1. How to Select a Good Product Owner?

Before starting training, it is important to select the right person.
Some essential qualities and skills she should have are:

**a) Business Vision**
A PO must have a deep understanding of the market, customers, and strategic goals of the company. He or she is not just a “backlog manager,” but a product strategist.

**b) Communication Skills**
The PO needs to clearly articulate the product vision, negotiate priorities, and mediate conflicts between stakeholders and the development team.

**c) Decision-Making Ability**
Prioritizing demands in a dynamic environment requires quick judgment and the courage to say “no” when necessary.

**d) Basic Technical Knowledge**
While the PO does not need to be a developer, he or she must understand the technical challenges to make realistic decisions.

**e) Ownership Mentality**
A good PO acts like a CEO of the product, taking responsibility for the success or failure of what is being delivered.

# 2. How to Train Product Owners

Once selected, the PO needs ongoing training to remain effective. Some strategies include:

**a) Mentoring and Shadowing**
Allowing the new PO to observe an experienced PO in action helps them understand the dynamics of the role in practice.

**b) Training in Agile Methods**
Courses in Scrum, Lean, Kanban and other methodologies and frameworks help the PO understand their role in the agile ecosystem.

**c) Professional Scrum Product Owner (PSPO) Certification**
PSPO, offered by Scrum.org, is one of the most respected certifications for POs. It validates knowledge in:
-Definition and management of the product backlog
-Value-based prioritization
-Collaboration with stakeholders
-Product metrics and ROI

Certification ensures that the PO understands Scrum best practices and is prepared to make strategic decisions.

**d) Continuous Feedback and Retrospectives**
Like the Scrum team, the PO must evolve based on regular feedback from stakeholders and the development team.

## 3. Why is PSPO Certification Important?

Many organizations question whether certification is really necessary.
The answer is: it depends. If the company is looking for:
-Standardization of knowledge among your POs
-External validation of the professional's skills
-Greater credibility with customers and stakeholders

So, PSPO is a valuable investment. However, certification alone does not make a good PO – practical experience and soft skills are equally essential.

# Conclusion

Selecting and training a Product Owner requires a focus on technical, business, and emotional skills. The PSPO qualification can excel in ensuring that the PO has the fundamental understanding to be a value maximizer. Invest resources in ongoing learning, mentoring, and recognized qualifications, but remember: a great PO is someone who combines theoretical knowledge, practical experience, and passion for the product.

# What kind of product owner are you? And how can you evolve?

The role of Product Owner (PO) is among the most complex and flexible in the agile environment, requiring a mix of technical skills, business acumen and communication skills. But what is your profile as a PO? And what can you do to progress in your professional career?

In this text, we will analyze the different types of Product Owners, identify areas that can be improved and establish a roadmap for career development.

## 1. Knowing Your Product Owner Profile

Each Product Owner has their own way of working, which is influenced by their skills, experience and the organization's environment. Here are some of the most common profiles:

**a) Tactical PO (Focused on Execution)**

Attributes: He prioritizes fast deliveries, works with stories well-defined user roles and ensures that the Scrum team has everything it needs to get the job done.

Advantages: Excellent at maintaining the pace of sprints and eliminating obstacles that could slow down the team.

Obstacles: You may end up leaving aside the strategic vision of the product, forgetting to align the backlog with long-term objectives.

**b) The Visionary (Strategic) PO**

Attributes: Sees the product globally, establishes clear plans and maintains a close connection with those involved.

Advantages: Has an exceptional ability to convey the product vision and ensure the team focuses on what is truly meaningful.

Obstacles: On some occasions, it can divert attention from implementation aspects, resulting in gaps in the backlog.

**c) The Hybrid PO (Tactical + Strategic)**

Attributes: Combines immediate execution with a long-term perspective, adjusting as needed.

Advantages: Adaptable and able to align team efforts with company goals.

Obstacles: May experience overload, trying to perform multiple tasks simultaneously.

**d) The Technical PO (With Background in Development)**

Attributes: Has a deep understanding of technology, which helps him evaluate and prioritize items based on their complexity and effects.

Advantages: Improves communication with developers and eliminates technical bottlenecks.

Obstacles: You may end up prioritizing technical solutions over user experience.

**e) The Commercial PO (Business Focused)**

Attributes: Has comprehensive knowledge of the market, customers and business performance metrics.

Advantages: Excellent at validating assumptions and ensuring a return on investment.

Obstacles: May not adequately account for technical or user experience difficulties.

## Improving as a Professional

To evolve in the role of Product Owner, it is essential:

Improve business skills: Know the sector, the market and the consumers.

Improve Communication: Cultivate effective communication skills to interact with stakeholders and teams.
The PO is, above all, a sieve of tastes, intentions and interests.

Prioritize with a focus on value: Learn to rank features and requirements based on customer value and business goals.

Make informed decisions: Improve analytical and decision-making skills to support the choices made.
**Seek feedback:** Request input from stakeholders, consumers, and teams to improve your PO performance

# Conclusion

The role of the Product Owner is essential in agile teams, and professional development requires specific skills and competencies. By understanding the different types of Product Owners and cultivating the traits of an effective Product Owner, you can increase your professional effectiveness and deliver valuable products to customers and stakeholders.
The book "The Professional Product Owner" - *Don McGreal and Ralph Jocham* offers valuable insights on how to improve as a PO and achieve success in agile teams.

# Should stakeholders participate in backlog refinement?

Scrum is a lightweight framework that can be adapted to the processes and needs of each company and product (not to be confused with Escrumlacho).

In teams that work with complex and legacy systems, the participation of stakeholders who work with the product on a daily basis,
from the devs who can look into the guts of the systems, and the PO who created the vision and goal of the product, it has everything to make backlog refinement sessions more productive, helping the PO to be more precise in creating and prioritizing stories.

Another advantage of this strategy is that future good P.Oÿ can be identified among current stakeholders.

# You can change the sprint stories backlog?

A common misconception about Scrum is that after planning you cannot change anything in the Sprint backlog. I am not talking about the details and the breakdown of PBIs into tasks.

I'm talking about exchanging PBIs.

Many call PBIs stories.

Living in Brazil, a country where the economy, laws, and regulations change daily, I have experienced many situations where the flexibility of Scrum was put to the test.

What cannot be done is for Sprint's objective to be disrespected, that's all.
Items may no longer make sense in the middle of a Sprint for a variety of reasons (a new government regulation, a company acquisition, the loss of a customer).


It's obvious that no one should sit idly by because of this, the team just needs to reach a consensus (PO/Devs) on which backlog item can replace the one that was cancelled.

I've been in Sprints that were canceled by the P.O. because the entire Sprint stopped making sense.

When something urgent comes up, it's never a big story, or worse, an epic. New epics that are so urgent would be very rare market opportunities that they would even justify canceling the current Sprint, but in 13 years this has never happened to me.

We have had to deal with small urgent items, sometimes the team was able to accommodate the new item without having to cancel anything, sometimes we take an item that was not essential and take it out of the Sprint, sometimes we agree to create technical debt.

**Conclusion: The** important thing is common sense.

# The importance of backlog refinement meetings.

In the world of agile, backlog refinement meetings are essential to ensuring teams are aligned and focused on the right priorities. Much underrated, these meetings play a crucial role in preparing the backlog for future sprints, ensuring items are clear, prioritized, and ready to be worked on.

Backlog refinement is the continuous process of reviewing, updating, and prioritization of items in the product backlog.
Items must be clear, concise, and aligned with the product and business goals. Without a well-refined backlog, teams can end up working on low-priority or poorly defined items, which can lead to wasted time and resources.


Nothing else is estimated in the backlog, especially story points, especially in the backlog, this is a waste of time and money.

The ideal frequency for backlog refinement meetings can vary depending on the size of the team and the complexity of the project.
However, a good rule of thumb is to hold these meetings once a week. This allows the team to keep the backlog up to date and aligned with current priorities, without overwhelming team members with excessive meetings.

Some teams wait until the planning ceremony to refine their backlog, but this is a bad idea. Planning is a critical time to define the sprint goals and plan the work to be done. If the team tries to refine the backlog during the planning ceremony, they may end up spending too much time discussing details of items that are not ready to be worked on, which can delay the start of the sprint.

On the other hand, the review is a great opportunity to refine the backlog. During the review, the team presents the work done in the previous sprint and receives feedback from stakeholders. This is an ideal time to discuss priorities, clarify doubts and update the backlog based on the feedback received.

## Conclusion

In short, backlog refinement meetings are critical to success in agile development. Holding these meetings once
per week can help keep the backlog up to date and aligned with current priorities. Additionally, it is important to avoid refining the backlog during the planning ceremony and use the review as an opportunity to refine the backlog.

With a well-refined backlog, teams can work more efficiently and effectively, delivering value to customers and stakeholders.

# The Daily Scrum revolutionized software development.

You may not even like her.
But **Daily Scrum** revolutionized the software industry.

I started my career at a software house. One of the first clients I worked with was also hiring another company. We used “prototyping” (I called it the DOITNOWBRO method), while the other software house used the waterfall method.

In other words, I delivered something new almost every day, the guy from the other software house would arrive, go to a remote corner, and close himself off in his own world, at a time when firewalls didn't exist.

We were two “lonely cowboys”, what changed was the focus, the quality
and the methodology.

Well, four months later he resigned. And guess what: he had done practically nothing, and what he had done was of no quality whatsoever. Soon, the client gave us exclusivity; he said we were too informal, but that we were the only ones who could deliver results.

Agile methodologies and frameworks, the most successful being Scrum, came to fill this gap between prototyping and Waterfall, but also to provide transparency, not let anyone get stuck (even providing assistance), enable inspection, and put an end to
big bad surprises.

As described in the Scrum Guide, the purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as needed, adjusting the work planned for the coming days.

The Daily Scrum is purposely short, it serves to discuss the progress of the sprint, nothing prevents other meetings from taking place during the day, if necessary, but having few extra meetings is a sign of a mature team.

Fifteen minutes are equivalent to 0.0104166666666667 of a day, they are not missing anything in your time. If you are an anti-daily, you don't want to "give satisfaction to anyone", "HELLO!!!", nowadays, lonely cowboys invariably end up fired, and rest assured, they are never the ones promoted.

One of the reasons for the Daily is to give the opportunity to give and ask for help, in Scrum we row in the same direction all the time.

I know programmers who complained about sitting all day and having to stand for 15 minutes.

**Conclusion: The** Daily circulates information, increases team confidence and cohesion and contributes to everyone's professional improvement. It also reduces the chance of delays during the sprint.

# Techniques for Backlog prioritization.

Backlog Prioritization: Techniques to Maximize Project Value
Backlog prioritization is a fundamental step in agile project management. It ensures that the most important and valuable tasks are completed first, thereby maximizing return on investment (ROI) and customer satisfaction. In this article, we'll explore some of the most effective techniques for backlog prioritization, including Planning Poker, the MoSCoW method, and other useful approaches.

## 1. MoSCoW Method

The MoSCoW method is a simple yet effective prioritization technique that categorizes tasks into four groups:

Must-Haves: Critical tasks without which the project cannot be considered a success.

Should-Haves: Tasks that are important but not essential to the success of the project.

Could-Haves: Tasks that are desirable but not necessary.
Won't-Haves: Tasks that are not a priority or can be discarded.

Advantages: Easy to understand and apply, helps you focus on the most important tasks.

## 2. Eisenhower Technique

The Eisenhower Technique, also known as the Eisenhower Matrix,
is a prioritization tool that helps you distinguish between urgent and important tasks. It divides tasks into four quadrants:

Urgent and Important: Tasks that need to be done immediately.
Not Urgent, but Important: Tasks that are important to the project but do not have an imminent deadline.
Urgent but Not Important: Tasks that can be delegated.
Not Urgent and Not Important: Tasks that can be eliminated.

Advantages: Helps you manage your time effectively, focusing on tasks that really matter.

## 3. Kano Model

The Kano Model is a prioritization technique that classifies product features into
three categories:

Basic (Expected): Features that are expected by the customer.
Performance (More is Better): Features that improve customer satisfaction the more they are offered.
Excitement (Surprise): Unexpected features that provide
great satisfaction.

Advantages: Helps you better understand customer needs and prioritize features that will have the greatest impact on customer satisfaction.

## 4. Value vs Complexity

This technique involves plotting tasks on a graph based on their business value and complexity. Tasks that offer high value with low complexity are prioritized first.

Advantages: Easy to visualize and helps the team focus on tasks that offer the best return on investment.

## 5-Planning Poker no longer makes sense

With the advent of Kanban for Scrum (see PSK1 certification), nothing is estimated in story points anymore, real data such as Cycle Time and
throughput.

## Conclusion

Backlog prioritization is crucial to the success of any project.
Using techniques such as Planning Poker, MoSCoW method, Eisenhower Technique, Kano Model and Value vs Complexity, teams can ensure they are working on the most important and valuable tasks.
Each technique has its advantages and can be chosen based on the specific needs of the project and team. By prioritizing effectively, teams can maximize project value, improve customer satisfaction, and achieve business goals.

# Dealing with toxic members and cliques on the Scrum team.

It is essential to understand the signs before acting:

**Toxic members:** Attitudes such as frequent criticism without presenting alternatives, passive-aggressive resistance to innovations, or rumors that compromise group harmony.

**Cliques:** Groups that deliberate independently,
leave other members out during conversations or establish an "us versus them" dynamic within the group.

The Scrum Master needs to pay attention to the interactions that occur in events such as the Daily Scrum and Sprint Retrospective. For example, if some participants monopolize the discussions while others are left to the side,
margin, or if choices are made outside of formal processes, it is time to intervene.

## 1. Creating a Safe and Inclusive Environment.

Having an environment where people feel emotionally safe is essential for healthy teams. The Scrum Master can help with this in several ways:

- Promoting open conversations: During retrospectives, encourage everyone to talk about behaviors and obstacles without fear of judgment. Questions like “What is hindering our collaboration?” help stimulate reflection in a constructive way.

- Establishing rules of coexistence: Together, define simple agreements, such as respecting all opinions or avoiding side conversations that could harm the group's focus (a hard task in times of home office).

- Valuing different ideas: Ensure that everyone has the opportunity to participate, using techniques such as round-robin, where each person speaks one at a time, so that all voices are heard.

## 2. Dealing with Toxic Members.

Direct conversation: Talk privately with the person, using examples
Specific questions, such as: "I noticed that during planning, you interrupted John three times. How can we improve this?" It is important to avoid accusations and focus on the impact that the behavior has on the team.

Offer help: Sometimes toxicity can be related to frustrations that have not yet been resolved.
How can a Scrum Master help remove personal or professional obstacles?

Setting clear boundaries: Most of the time it has been possible to improve the situation, but if the behavior continues, it is important to involve the leadership, such as HR or managers, to take the necessary actions, always following the organization's policies.

## 3. How to avoid internal conflicts.

Internal conflicts often arise due to a lack of trust or different goals among team members. To avoid these situations, the Scrum master can:

- Encourage activities that promote collaboration, such as reviewing each other's code (and taking responsibility for the result of the code in production),

group work or workshops involving people from different areas.

- Reinforce the common goal: remind the team of the product vision and how each person has an important role in achieving that goal.

- Questioning decisions taken in isolation: if a subgroup decides something without consulting the rest of the team, bring it up for a larger conversation (preferably in the Sprint Retrospective), asking, "How can we ensure that everyone participates in this discussion?"

## 4. Real situations that I experienced.

In 15 years I have been through several situations involving acidic members and cliques.

Seniors who were wrong, but defended what had always been this way.
Juniors who were wrong and used ageism as a form of defense.

Cliques involving a “dominant” who used his influence to overthrow another member and thus benefit from the situation (salary).
Persecution of members for political reasons.
Cliques aiming to overthrow the Scrum Master because they didn't want changes in the team culture.
Members who had persecution mania.
Members who, feeling frustrated or undervalued in the company, tried to sink the ship with everyone else on board.
Members of the three roles (Scrum masters, devs and POs) who had technical deficiencies but did not accept any type of criticism.

## 5. Prevention: Cultivating a Strong Agile Culture.

The most effective approach is to anticipate and avoid difficulties before they arise.

-Apply the principles of Scrum: The Scrum Master must exemplify respect, clarity and boldness in his/her attitudes.

-Appreciate group victories: Value collaboration between members, rather than focusing solely on personal achievements.

-Invest in ongoing education: Courses on non-violent communication or conflict management improve team maturity.

## Conclusion:

Having toxic team members and cliques poses delicate but surmountable challenges. The Scrum Master needs to act as a supportive leader, combining understanding and determination to rebuild trust and foster collaboration. By creating a safe space for open conversations, fostering sincere dialogue, and reinforcing a common purpose, it is possible to convert conflicts into opportunities for team development. It is important to remember that a cohesive team is not made up of an absence of disagreements, but rather the ability to resolve them with respect and a focus on the shared goal.

Page left intentionally blank

# The different possible configurations of the Scrum team.

A Scrum team must have between 3 and 10 people, and must have a PO, a Scrum master and at least one developer.

A product should have only one product owner.
A Product Owner can be on more than one team and take care of more than one product.

A Scrum master can be on more than one team.

A Developer can be on more than one team, although this is very rare.

A Developer can jump from team to team, every sprint (in the case of legacy technologies, with difficult labor, for example).

A QA is nothing more than a developer who understands a lot about testing. In my opinion, a QA should be more of a test coordinator than a tester.

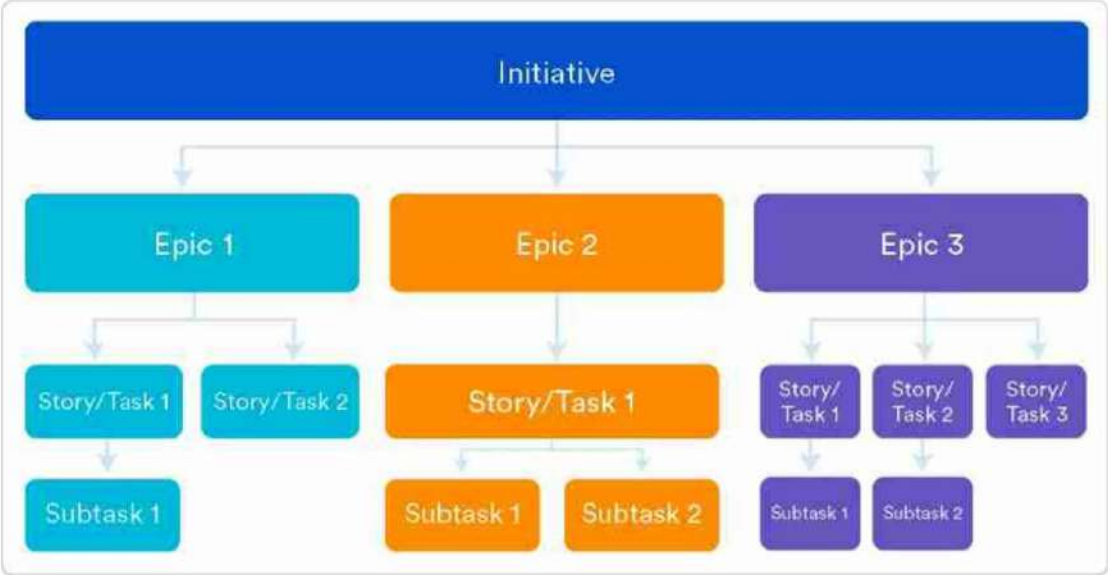
Conclusion: It may, but that doesn't mean it's ideal, companies have different sizes and needs, and Scrum adapts to all of them.

# User stories, acceptance criteria and tasks, taking the Java Pet Store as an example.

In software development, the ability to tell stories effectively is essential to transforming vague requirements into concrete solutions. Inspired by the teachings of Scrum.org, we will explore how to create stories and break them down into manageable tasks, using the example of the Java Pet Store.

What is Storytelling in the Context of Software Development?

In the context of software development, storytelling refers to the practice of capturing requirements or desired system functionality in the form of user stories. These stories are short narratives that describe something the user wants to accomplish within the system. They
are fundamental to Agile and Scrum, as they help maintain focus on the value delivered to the user.




## Creating Effective Stories

To create effective stories, we must follow some basic principles:

**Stories should be user-centric:** They should clearly describe who the user is, what they want to accomplish, and why.

Use the standard format: Use the format "As a [user], I want to [accomplish something] to [achieve some goal]."

Keep them small and manageable: Stories should be granular enough to be completed within a sprint.

**Example: Java Pet Store**

Let's consider the example of Java Pet Store, a demo application that simulates an online pet store. A story for this application could be:

Story: "As a store customer, I want to view the details of a specific product to decide whether to purchase it."

**Acceptance Criteria:** Customer can view product details including name, description, price and image.

Product details are displayed correctly for different types of products (dogs, cats, etc.).

Breaking Down History into Tasks

Now, let's break this story down into technical tasks that developers can perform:

**Task 1: Create a product detail page**

Description: Develop the interface of the page with product details.

Acceptance Criteria:

The page should display the product details correctly.

The page must be responsive and adapt to different devices.

**Task 2: Implement product details loading logic**

Description: Implement the business logic to load the product details.

Acceptance Criteria:

Product details are loaded correctly from the database.

The loading logic is efficient and does not cause performance issues.

**Task 3: Add an image carousel**

Description: Add an image carousel for products.

Acceptance Criteria:

The image carousel is displayed correctly on the product detail page.

The image carousel is responsive and adapts to different devices.

**Task 4: Test the functionality**

Description: Write unit and integration tests to ensure that the functionality is correct.

Acceptance Criteria:

Unit and integration tests are written and pass with success.

The functionality is tested and approved by the Product Owner.

## Conclusion

Storytelling in software development is not just a technique for capturing requirements; it is a way to ensure that development is always focused on the user and the value they receive. By following the Scrum.org principles and breaking stories down into manageable tasks, as demonstrated with the Java Pet Store example, we can ensure that we deliver high-quality software that meets the needs of our users.

# Integrating Design Thinking into Scrum.

## How Creative Sessions Can Enhance Your Backlog Refinement.

In the agile product development process, Scrum and Design Thinking are commonly considered complementary frameworks. Scrum provides a framework for iterative and incremental delivery, while Design Thinking adopts a user-centric perspective, encouraging empathy, idea generation, and rapid prototyping.

But how can these two frameworks be combined in the routine of a Scrum team? A common practice is to have a developer focused on UX and Design Thinking, but there is a growing consensus that all developers and the Product Owner (PO) should improve these skills. In addition, Design Thinking sessions can (and should!) take place during the sprint, as well as backlog refinement — and, in many situations, they can even be seen

as part of backlog refinement.

**Why Should Everyone on the Team Understand Design Thinking?**
Having a UX expert on the team is valuable, but limiting this expertise to a single person can create a bottleneck. If all developers and the PO have notions of Design Thinking, the team gains:

Better user understanding, resulting in more relevant solutions.

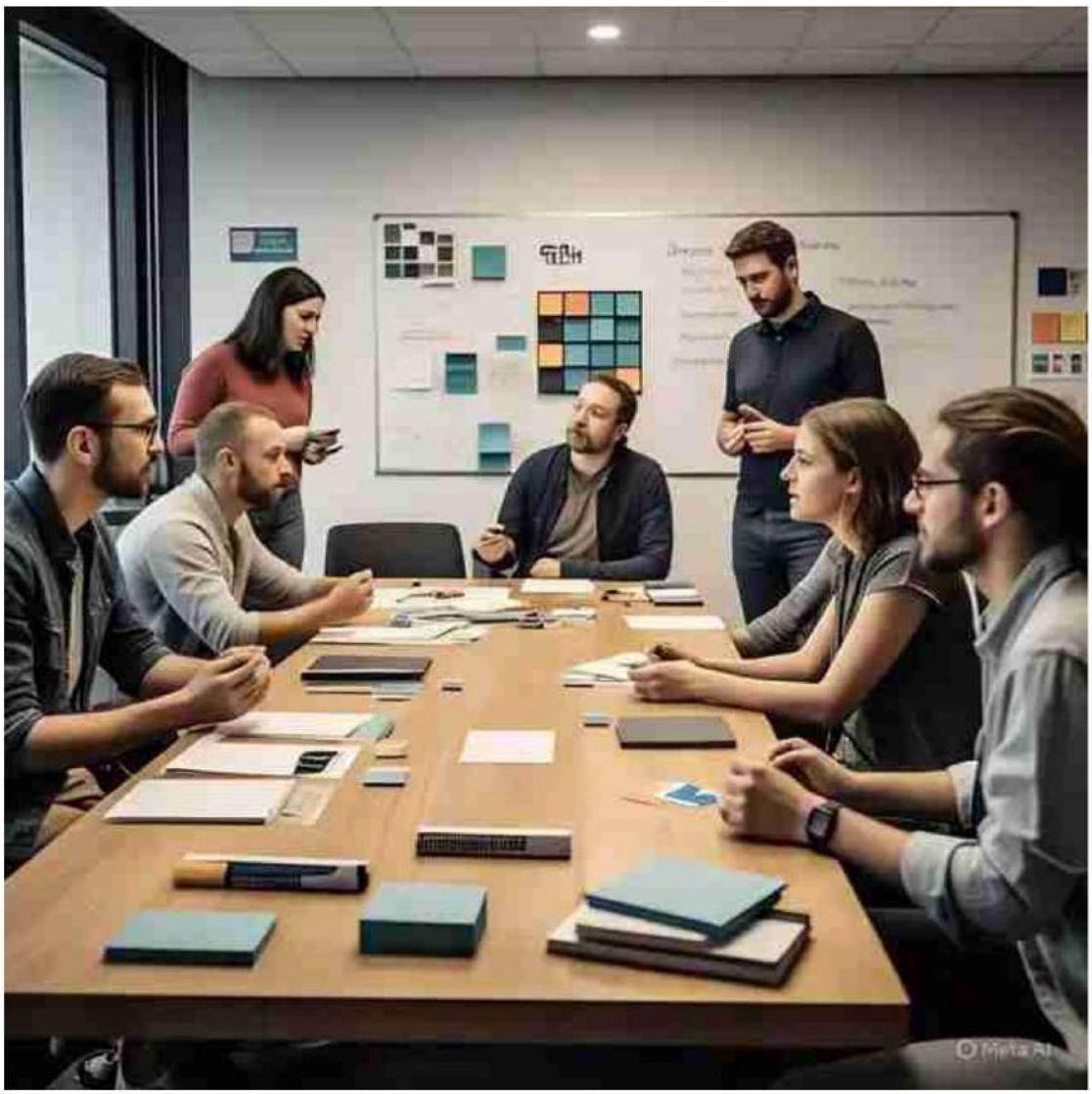
More efficient collaboration between development and design.

More refined backlog, with clearer and more validated user stories.

The PO, in particular, benefits from incorporating Design Thinking techniques as they can prioritize demands based on real evidence rather than just assumptions.

Design Thinking Sessions During the Sprint: Scrum is an adaptable framework that allows meetings beyond the official ceremonies. Just like
backlog refinement, a best practice, Design sessions

Thinking can also be integrated into the sprint flow in a natural way, enriching the team's process and results.



It is important to highlight that the presence of stakeholders is always welcome in any backlog refinement meeting, especially when it comes to design thinking sessions,
These sessions may include:

- Rapid brainstorming to explore solutions before development.

-Usability testing with real users to validate hypotheses.
- Low-fi prototyping to align expectations before coding.
Can These Sessions Be Considered Refinement?

Yes! Some experts argue that ideation and validation activities are part of backlog refinement because they help to:

Clarify requirements before transforming them into technical tasks.

Validate assumptions before committing development effort.

Align expectations between PO, designers and developers.

**Jeff Patton,** creator of User Story Mapping, argues that refinement is not just about breaking down tasks, but about deeply understanding the problem. Similarly, **Jake Knapp,** author of Design Sprint, shows how short ideation cycles can avoid rework in development.

**Conclusion:** Design Thinking is Not a Separate Process, But Part of the Agile Flow

The integration of Design Thinking and Scrum is not only possible, but should be encouraged. Instead of treating UX and development as isolated steps, agile teams can incorporate creative sessions into the sprint, treating them as part of the continuous refinement of the backlog.

## Strategies to ensure that QA does not become a bottleneck in the Sprint.

First of all, it should be said that Scrum does not recognize differences between developers, QA is considered a developer.
The testing discipline must be respected, it is as deep as systems development, but the ideal is for each developer to improve their testing skills, learning about integration and functional testing is a must.

If the team has a dev specialized in testing (read QA), then this
It should act as a test advisor, it should help in the development of the test plan, it should ratify the tests done by each dev, in addition to doing its own, but finding an error at this stage should be an exception, capable of being discussed in the sprint retrospective.

**QA is not a programmer's babysitter.**

Our DEV, who understands testing, can be responsible for setting up the automated testing pipeline as well.
Scrum is a framework for product development, not just software, a developer can be a wall painter, all his

Do devs program in Java, or in Oracle? No, so QA is a dev and that's it.

One dev testing another's work, before passing it on to our specialist testing dev (QA), usually yields good results.
Tests should be part of the definition of done and the definition of workflow, avoiding overloading QA and leaving everything to the last minute.

A good Scrum Master should know about WIP, workflow definition and Service level expectation to help the team with their definitions.
Having a company pay for a trial course on Udemy is a good strategy.

The Scrum.org DEV certification covers a lot of the area of testing.
Scrum is a framework that can be adapted to the company's reality and practices (this should not be confused with relaxed Scrum).
The more the PO understands the product, and is able to transform this into good stories, less need for stakeholders for UAT testing, but there is nothing stopping stakeholders from helping with testing, this is especially useful when the team works with legacy and complex systems.

It is very important that our test-savvy dev actively participate in the refinement of the backlog.

Although the review should not be seen as a gateway to sending value to production, it is expected that most of what was agreed upon will be demonstrated at the end of the sprint. It is not worth leaving everything for QA to check on the Thursday before the review, when it is often too late to fix anything.

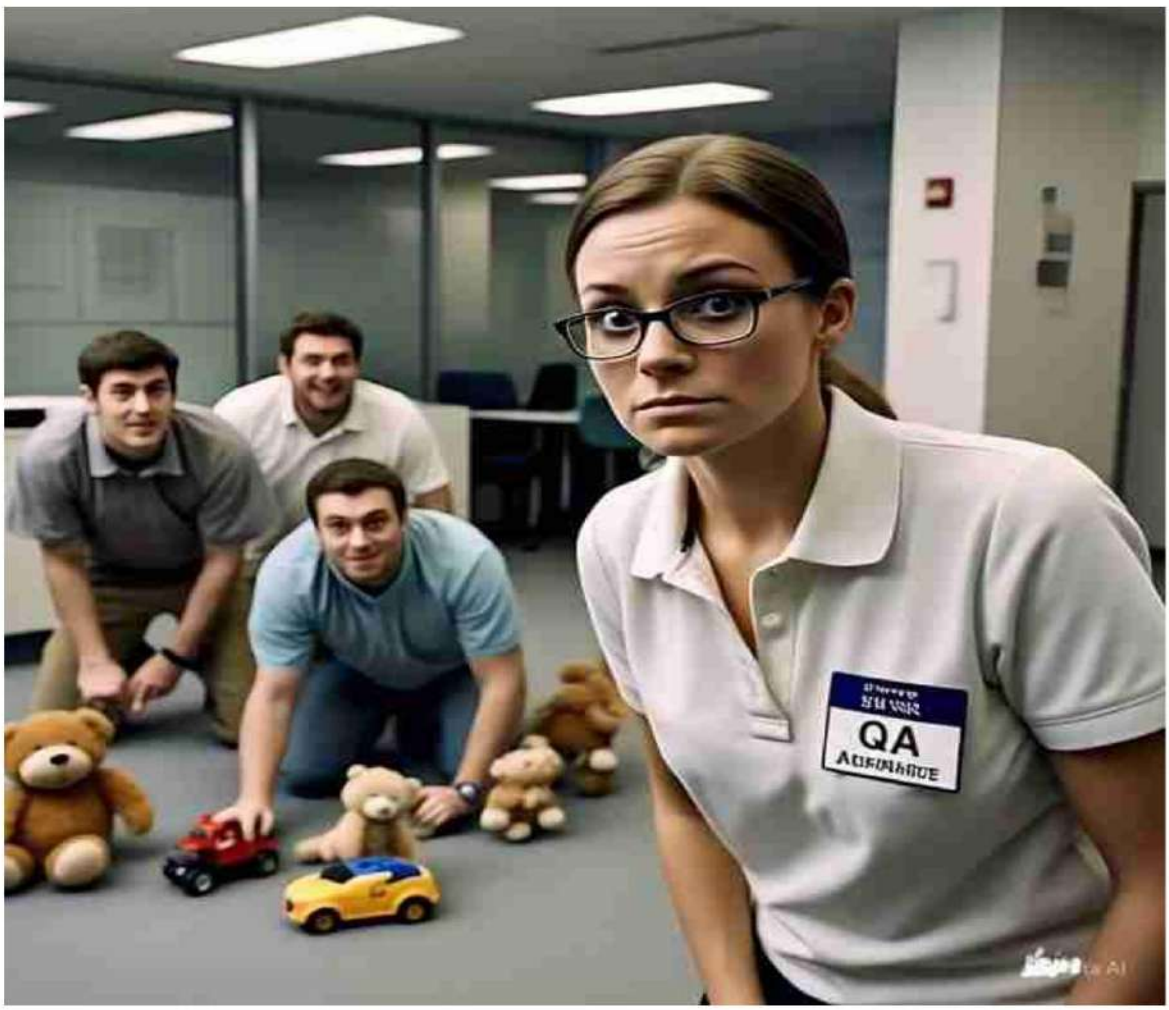
A defect is not a technical debt, it is a defect, if something is defective, it does not match the definition of done, and should not be demonstrated in the review.

Although continuous delivery reduces the pressure of delivery in the review, it does not if it should be done due to process failures

**Conclusion: QA** is not a developer's babysitter, everyone on the Scrum team
They must understand testing, the QA's role is to coordinate and ensure the quality of the tests, he must also get his hands dirty, but in a logical and coordinated manner.

# CAMPAIGN

# QA IS NOT A NANNY.



# Replacement in the Scrum team: Story points and Capacity are out, Throughput and Cycle time are in.

Since 2012, Ken Schwaber (probably inspired by Lean Kanban), one of the creators of Scrum, has questioned the validity of using metrics such as velocity within the Scrum framework.

In his article "Waste Not", he argues that spending time worrying about this metric is a waste that brings no real value to the development process.

This opinion becomes even more important when we think about Scrum teams that also use Kanban practices, where indicators such as throughput and cycle time have proven to be much more efficient for planning the work that will be included in the Sprint Backlog.


**Ron Jeffries,** the creator of story points, no longer uses them. If you are interested, search for Ron Jeffries "Story Points-Sorry".
Unfortunately, story point-based forecasts have become a crutch that, for a long time, helped to relieve pressure on Scrum teams. However, it is now clear that these estimates do more harm than good. Still, it is

It is difficult to convince many Scrum Masters, especially those who are not certified by Scrum.org, to abandon them in favor of adopting real data, such as cycle time and throughput.

In the current context, they are widely seen as unnecessary and, from a Lean perspective, can be considered waste. I can just imagine the shocked reaction of some Scrum Masters when reading this.

## The Fundamental Problems with Velocity

**Velocity:** A Misleading Metric
Velocity, traditionally measured in story points per Sprint, presents several problems:

**False precision:** Story points are subjective estimates that vary between teams and even between members of the same team over time.

**Invalid Comparisons:** As Ken Schwaber pointed out, velocity should not be used to compare teams, as it is a team-specific metric.

**Wrong focus:** Teams may start optimizing locally to increase velocity, to the detriment of delivering real value to the customer.

## The new approach

The Scrum Approach with Kanban: Throughput and Cycle Time
The Professional Scrum with Kanban (PSK) certification, offered by Scrum.org, takes a different approach, based on the flow principles of Kanban. In this method, two main metrics are

highlights:

**Throughput:** Quantity of Items Delivered
Defined as "the number of work items completed per unit of time", throughput provides:

**Objective measurement:** Based on actual items delivered, not estimates.

**Greater accuracy:** Historical throughput patterns allow for slightly more accurate predictions (remember, there is no such thing as a crystal ball).

**Simplicity:** Easy to measure and understand by all stakeholders.

**Cycle Time:** Time to Complete an Item
Cycle time measures how long it takes for an item to flow from start to finish in a team's workflow. Its advantages include:

**Bottleneck identification:** Shows where work is getting stuck.

**Estimated Delivery:** Allows you to estimate when future items will be completed

**Continuous improvement:** Reducing cycle time is a clear process improvement objective.

## Service Level Expectation (SLE): The New Benchmark for Sprint Planning

1. An important concept we learn in preparation for the PSK certification is Service Level Expectation, or SLE. It is basically
a forecast of how long it should take for a given task to move through the entire process within the Scrum team's workflow.

In practice, SLE works as an alternative to traditional estimates during Sprint planning.
Rapid Decision: Instead of long estimation sessions, the team simply asks: Can this PBI be completed within our SLE?

**Focus on right-sizing:** Items that are too large for the SLE are split before entering the Sprint Backlog (decomposition of stories into tasks).

**Continuous adaptation:** The SLE is refined based on real historical cycle time data

Implementando throughput-Driven Sprint Planning
Based on Scrum.org's materials for the PSK certification, here is how a team can implement throughput-based planning:

**Visualize the flow:** Create a Kanban board that clearly shows the stages of the workflow. (search for definition of Workflow Scrum.org).

**WIP Limit:** Set work in progress limits for each stage.

**Measure historical throughput:** Analyze how many items were completed in the last Sprints.

**Define SLE:** Set a realistic expectation based on historical cycle times.

**Plan with throughput:** In Sprint Planning, select items until you reach historical average throughput.

Teams that take this approach often say that:

**Greater predictability:** Throughput is more stable than velocity in story points.

**Less time spent on estimation:** SLE-based decisions are made in seconds.

**Better value stream:** WIP limits prevent overhead and improve quality.

**Transparency:** Metrics based on real data build trust with stakeholders.

**Conclusion:** Ken Schwaber, back in 2012, pointed out that metrics such as velocity could be considered waste in agile processes. The approach presented in the Scrum.org PSK certification, with an emphasis on throughput and cycle time, emerges as an alternative that is more aligned with the principles of empiricism and continuous improvement that are characteristic of agile thinking. For teams that have already mastered the fundamentals of Scrum, incorporating Kanban practices can represent a natural advancement in their agile journey. This adoption promotes greater predictability, workflow optimization and, above all, delivery.

consistent customer value. Teams interested in exploring this approach in more depth can take advantage of the resources offered by Scrum.org for the Professional Scrum with Kanban certification. This certification provides a clear guide to integrating these practices while maintaining the essence of the core Scrum principles.

# Wip, Cycle time, Lead time, Throughput , Lei de Little.....

This is the only boring article in this book, this article has no intention of preparing anyone for PSK1, just giving a brief overview of the subject and instigating the reader's curiosity.

I am convinced that once you assimilate everything, you will find it much easier and more agile to deal with these indicators than if you just remain in the universe of imagination, reflecting on story points, capacity and speed.

Na **PSK1** (Professional Scrum Kanban Level 1), **Leading** e **Lag**

**Indicators** are metrics used to monitor and improve the flow of

work in the context of Scrum combined with Kanban. Let's understand

each one and its application in Scrum ceremonies:

When we use Scrum together with Kanban, the so-called leading, or anticipatory, indicators help predict what might happen in the future. Lagging, or lagging, indicators show what has already happened. Examples of leading indicators include "Work in Progress" (WIP) and "Work item age". Lagging indicators include
"Cycle Time" and "Throughput" (productivity).

## 1. Fundamental Concepts

### (A) Cycle Time vs. Lead Time

## Definitions:

**Cycle** Time: Active time to complete an item (e.g. from "In Progress" to "Done")4 days (coding + testing)

**Lead** Time:Total time from request to delivery10 days (including waiting in backlog)

**Work Item** Age: refers to the amount of time a work item (task, bug, etc.) remains in a state of flow on the Kanban board.

# Application in Scrum with Kanban:

**Lead Time** is essential to understanding how quickly value is delivered.
It measures the total period from the creation of a task to its completion, providing meaningful data on the effectiveness of the workflow.

**Cycle Time** helps the team to optimize the flow, as it: Helps in measuring efficiency, identifying bottlenecks, optimizing processes and helps in defining deadlines.

**(B) Throughput (Delivery Rate)**

Definition: Number of items completed per unit of time (e.g. 5 items/sprint).

Formula:
Throughput = Items completed / Time (e.g. per sprint)

**(C) Little's Law: The Mathematical Basis of Flow**
WIP = Throughput × Cycle Time
Example:

If a team has Throughput = 10 items/sprint and Cycle Time = 2 days/item, then:

Optimal WIP = 10 × 2 = 20 "work days" in progress

Therefore, the WIP is limited as follows: If the sprint has 10 working days, the maximum WIP should be 2 items in parallel (20 days / 10 days).

**2. Defining Workflow and WIP in Scrum with Kanban**
(A) Basic Workflow (Example of Kanban columns)
Refined Backlog In Development In Test Ready to Deploy Done ÿ

**Important policies:**

Clear Definition of Done (DoD) for each stage.
WIP limits to avoid overload.

**(B) How to Calculate WIP at the Start of a Sprint?**

Factors that impact: -Holidays:
Reduce working days.

-Holidays: Reduce team capacity.

-Practical Example:

Team: 5 devs + 1 QA (I know he is a dev) = 6 people

Sprint: 2 weeks (10 business days), but there are 2 holidays and 1 dev on vacation.

Adjusted business days:

Total days = 10 - 2 holidays = 8 days

Capacity = 4 devs (since 1 is on vacation) + 1 QA = 5 people

Total capacity = 8 days × 5 people = 40 "working days"

If the average Cycle Time is 5 days/item:

Maximum WIP = Capacity / Cycle Time = 40 / 5 = 8 items

-Suggested limits:

Development: 4 items (1 per dev)

Test: 2 items (avoid bottleneck in QA) - This shows the importance of developers learning about these.

## 3. Critical Scenario: QA as a Bottleneck (and Its Impact on Flow)
## (A) The Problem

Team with only 1 QA for 4 devs.

Workflow without clear policies: Devs "push" items for testing without prior verification.

Result: Huge

queue in "In Testing" (WIP explodes).

Cycle Time increases (Little's Law in action).

(C) PSK Based Solutions I Limit WIP to "In Test" (e.g. maximum 2 items).

Include tests in the devs' Definition of Done (e.g. "No item goes to test without unit tests").

Cross-training: Devs should learn basic testing.

## 4. Common Mistake: The problem of devs who don't test and think that QA is their babysitter.

This violates the Scrum principle "Done = Potentially Shippable".
Creates over-dependence on QA.
Effect on WIP: Items get "stuck" in testing, increasing Cycle Time.

## How to fix:

Include tests in the developers' workflow:
In Development Unit/Integration Testing Review In Testing (QA) Done ÿ

Define clear policies:
"No item goes to QA without 90% test coverage."
User pair testing (dev + QA).

## 5. Tools to Visualize and Improve Flow

Cumulative flow diagram. Its main function is to graphically represent the progress of activities, allowing a clear visualization of the progress of the work and facilitating the identification of possible bottlenecks in the process. It makes it possible to monitor the progress of tasks over time, displaying the number of items in each phase of the process, from start to finish.

**WIP: Work in progress**

Bottlenecks: Columns that get stuck, cannot receive more items, as they have stopped items, and would violate the WIP limitation rule (e.g.: "In Test").

(B) Cycle Time- **Control Chart**

(C) Aging Work in Progress
Shows items "aging" in the flow (e.g. a bug stuck in testing for 3 days). Can be viewed through the **Aging WIP Chart.**

For a more detailed analysis, the best article (in English) that I know of that illustrates each of these graphs, as well as their applications in the various ceremonies, is the following:

https://www.scrum.org/resources/blog/4-key-flow-metrics-and-how-use-them-scrums-events

The book (in English) that helped me a lot to get the PSK1 was:
https://www.amazon.co.uk/Applying-Scrum-Kanban-Pointless-Book/dp/B09HFZCJFT

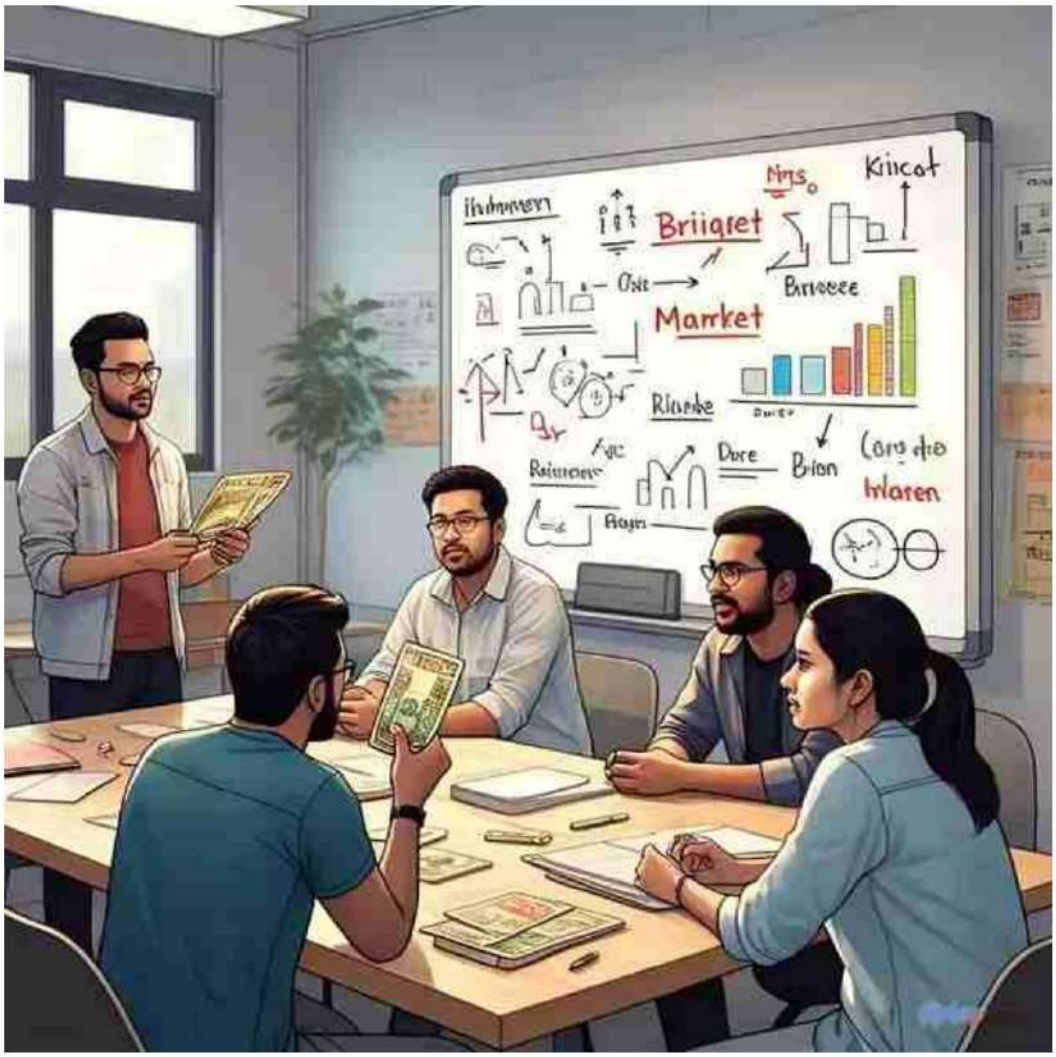
In my opinion, the PSK1 test has a much higher level of difficulty than the PSM1, making reading the book above mandatory.

# Dynamics aimed at strengthening Scrum values.

The Scrum Values are: Commitment, Courage, Focus, Openness and Respect

The Scrum Master is a key advocate for the Scrum principles of Commitment, Courage, Focus, Openness, and Respect. These principles are vital to high-performing agile teams, but they require deliberate care to uphold. When ignored, they can result in disagreement, decreased productivity, and lack of alignment. In this article, we will discuss practical strategies that the Scrum Master can implement to reinforce each of these values.

promoting a positive change in team culture.



## 1. Commitment

Commitment concerns the team's commitment to common objectives and service excellence, respecting the agreements reached.

"Commitment Map": At the beginning of the Sprint, the team draws a "map" with the Sprint Goals at the center. Each member writes on post-its: “How can I contribute?” and “What do I need from others?” Everyone shares answers and negotiates support. This makes individual and collective expectations explicit, creating a visual pact of accountability.

"Accountability Check-in": Every day after the Daily Scrum, set aside 5 minutes for a quick round of "What did I promise yesterday and deliver? What will I promise today?". Record the commitments on a board shared. Constant transparency reinforces mutual trust and a culture of accountability.

"Commitment Contract": Create a contract that all team members sign, committing to work as a team and respect the established agreements.

## 2. Courage

Courage involves the ability to face difficult challenges, give honest feedback, and try new paths.

“The Elephant in the Room”: In a feedback meeting, distribute anonymous slips of paper. Each person writes down an “elephant” (unspoken problem). The slips are then randomly drawn and discussed by the group, focusing on solutions. Initial anonymity reduces fear of retaliation by encouraging difficult truths.

"Experimenter's Challenge": The team chooses one "bold experiment" per Sprint. At the end, they discuss: "What did we learn, even if we failed?" This normalizes risk and highlights that courage is about learning, not perfection.

“Chaos Master”: Designate one team member to be the “Chaos Master” for each Sprint. This person is responsible for identifying and resolving critical issues, promoting courage and proactivity.

## 3. Focus

Focus involves prioritizing activities that add direct value to the Sprint goal while minimizing distractions.

"Distraction Hunting": In a meeting, list all the side activities of the team. Classify them as "Essential" or "Noise". Negotiate the elimination of at least "noise". This enables the team to protect its collective time.

"Collective Pomodoro": On busy days, the team works in synchronized 25-minute blocks. After each block, a quick shared break. The synchronized rhythm reduces multitasking and creates focused flow.

"Priority Board": Create a board that lists the team's priorities. This helps keep the focus on the most important activities.

## 4. Openness

Openness involves transparency about progress, obstacles, and ideas, creating a safe space for vulnerability.

"Show & Tell Mistakes": In the Retrospective, each member shares a mistake they made in the Sprint and how they fixed it. You celebrate the learning, not the guilt. This demystifies failure and encourages innovation.

"Marketplace of Ideas": Create a virtual "marketplace" where everyone posts ideas for improvements. Each person "invests" points in 3 other people's ideas. The ones with the most votes become actions. This democratizes creativity and shows that all voices matter.

"Feedback Session": Set aside time for team members to share feedback with each other. This promotes openness and transparency.

5. Respect

Respect involves valuing the contributions, abilities and limits of each member, promoting empathy.

"Feedback Gift": At the end of the Sprint, each person writes to another member: "A talent of yours that I admired was..." and "An idea for

you explore is...". Share out loud. This builds mutual recognition and respectful criticism.

"Colleague Persona": In pairs, each person interviews their partner about: "How do you prefer to receive feedback?", "What demotivates you?". Then, they introduce each other to the team. This promotes understanding of individualities and avoids conflicts.

“Thank Yous”: Take time for team members to thank each other for their contributions. This promotes respect and gratitude.

## Conclusion

Scrum principles are capabilities that need to be developed.
The Scrum Master should act as a "gardener of agile culture", using dynamics of this type to cultivate seeds of trust and teamwork. The main sign of success is not speed, but rather the unity of the group. Start with a dynamic centered on values, adjust it according to the reality of the team and see how maturity

agile develops.

# We will have to cancel the sprint, what now?

The Scrum Guide is clear in stating that cancellation should only occur when the Sprint Goal becomes obsolete. But what exactly makes a goal obsolete? In practice, several external and internal factors can justify this decision:

## 1. Economic and Market Changes

Significant fluctuations in the economy or abrupt changes in market conditions can render planned work obsolete. For example:

An economic crisis that forces the company to reprioritize initiatives.
Changes in interest rates that affect the viability of certain products.

Exchange rate fluctuations.
New import and export taxes.

## 2. Corporate Transformations

Acquisitions, mergers or sales of companies can completely change strategic priorities:

Acquisition of a subsidiary that brings new capabilities and makes current work redundant.

Sale of part of the business that makes the product under development unnecessary.

Merger with another company that already has a similar solution.

## 3. Loss of Strategic Customers

When a key client backs out of a project.

Cancellation of a contract with a customer that represented a significant portion of revenue.

Change in regulatory requirements that invalidate planned work.

Competition that launches a superior solution, or one with a very low cost, making the current product obsolete.

## 4. Technological Changes.

Technical discoveries or limitations that arise during the Sprint:

Base technology becomes obsolete or insecure.

Discovery that the proposed solution is not technically feasible.

Emergence of new technology that offers better cost-benefit.

## 5. Legal or Regulatory Changes.

New laws or regulations that make it impossible for the product under development to be launched.

Jurisprudence that impacts the viability of the project.

Changes in compliance standards.

The Cancellation Process: Roles and Responsibilities
It is crucial to understand that in Scrum, only the Product Owner has the authority to cancel a Sprint. However, this decision is typically made after consultation with the team and key stakeholders.

## What can the Product Owner do?

If the product owner decides to cancel the Sprint, he or she must inform and clarify the reasons for the decision to the stakeholders and the Scrum team. In addition, it is important that the product owner dedicates himself or herself to organizing and gathering the resources needed to reorder the backlog, with a focus on the next sprint.

## What can the Scrum Master do?

The Scrum Master does not decide, but should advise, facilitate discussion about why the objective is obsolete, and help the team understand the reason and implications of cancellation.

## What can developers do?

The team must provide technical information about the current state of work, actively participate in replanning.

What to do after cancellation?
When a Sprint is cancelled, it is essential to act quickly to minimize the negative impacts. The recommendation is to start

immediately start a new Sprint to get back into the rhythm and readjust the calendar.

### 1. Plan a buffer Sprint

An effective approach is:

Create a Sprint that ends on a Friday (the size is irrelevant, as long as it is not longer than a month), it does not matter that a Sprint starts on a Wednesday, and not on a Monday as usual.

Focus on critical items that help realign the product.

Finishing on Friday allows the team to resume the normal cycle the following Monday.

### 2. Dealing with Existing Work

Complete and "Ready" items, i.e. those that satisfy the Definition of Done, should be reviewed and accepted if they still make sense.

Incomplete work should be returned to the Product Backlog (If it still makes sense).

## 3. Review if it makes sense

If there are items that have been finalized and still make sense, they should be shown in the review, otherwise it will just be emotional wear and tear and a waste of time for those involved.

### 4. Conduct a special retrospective

A retrospective focused on the cancellation (on the day of the cancellation, or the next day) helps to:

Identify lessons learned.

Check whether the team had any responsibility for the event, not aiming to punish anyone, but to close the doors to prevent this from happening again.

**5-Impacts and risks of cancellation** Canceling a Sprint is not a trivial decision. As Julio Roche from Deloitte points out, "Sprint cancellations consume resources and are often traumatic for the Scrum Team."

5.1-Impact on Team Morale.

Frustration over seemingly wasted work.
Feeling of instability and uncertainty.
Potential decrease in motivation.

5.2-Operational Impact.
Time lost in replanning Cost of stopping and
restarting work Potential accumulation of technical debt.

5.3. Impact on the next Sprint.
The impact on the ability to select items for the next Sprint (Sprint Backlog) will be different for each maturity level (more mature teams will be less affected), teams that still use story points (which is not appropriate) will also suffer more than teams that rely on cycle time and Troughput.

## 6-Best Practices to Minimize Impacts

## To reduce the negative effects of cancellation:

6.1. Clear and Transparent Communication.
Explain the reasons for the decision to all stakeholders.
Make it clear that this is an exceptional measure.
Reaffirm the commitment to the new objectives.

6.2. Agile Replanning.
Perform a new Sprint Planning as soon as possible.
Keep the team focused on the new objective.
Set clear and achievable goals.

6.3. Continuous Learning.
Using experience to improve resilience.
Document lessons learned.
Adjust processes to avoid similar situations.


## Conclusion:

Canceling a Sprint always brings financial and moral losses, but it should not be seen as something absurd, but rather as a learning experience, and sometimes, as a business risk.
The important thing is to focus on containment and realignment measures. Nothing replaces empiricism, I just gave some suggestions based on my experience, but your team can, and should, have ideas on how to deal with a situation as delicate as the cancellation of a Sprint.

**Page left intentionally blank.**

# When QA becomes Product Owner.

First, it is not better or worse to be a QA or PO, I am writing this article because I am tired of answering this question for QAs, managers and HR staff.

The primary mission of a QA is to ensure software quality. This responsibility requires a critical and meticulous eye, capable of identifying functional flaws, inconsistencies, and opportunities for improvement. Such attention to detail is a significant advantage in the PO role, especially during backlog refinement, requirements validation, and acceptance criteria development. In addition, their experience with exploratory testing teaches them to “think outside the box” — a crucial skill.
to anticipate risks and predict real user behavior.
It must be said that I know a QA who regretted the change and went back to being a QA, as a matter of vocation.
It's always good to remember that there is a career plan for QAs.

### 1. Proximity to the Development Cycle

QAs are part of the day-to-day operations of agile teams, monitoring everything from daily meetings to sprint deliveries. They have a deep understanding of Scrum's rituals, artifacts, and roles, which makes it easier to adapt to the PO's routines and responsibilities. This constant interaction with developers and Scrum Masters helps create a natural bridge between the technical and business areas — a central role of the PO.

### 2. Focus on the End User and Real Experience

Testing a product isn't just about validating requirements: it's about imagining how the user will interact with it. Successful QAs develop empathy for the customer and are able to detect friction in the experience before it even hits the market. This user-centric mindset is essential for any PO who wants to deliver continuous value.

3. Prevention and Continuous Improvement Mentality
QAs live the culture of continuous improvement, through retros, automations testing, and constant feedback. This critical and proactive vision helps in the evolution of the product and in the improvement of internal processes — characteristics that greatly contribute to the PO in prioritizing features and refining requirements.

Challenges in Transition: What Are QAs Still Missing?
Despite the advantages, QAs who want to move into the PO role face a significant learning curve. The main gaps are in the strategic, behavioral and leadership fields.

1. Strategic Vision and Value Orientation
While QAs focus on finding bugs and ensuring stability, the PO needs to define what to build based on business goals, market needs, and customer behavior. This requires a shift in focus from operational efficiency to strategic impact.

Many QAs still have difficulty seeing the product as part of a larger ecosystem, where decisions need to generate measurable value.

2. Stakeholder Management and Negotiation of Interests
The PO acts as a link between technical and executive areas. It is necessary to deal with multiple interests — customers, leadership, marketing, sales, support — and know how to say "no" based on data and priorities. This ability to influence and articulate visions is little explored by QAs in their traditional roles, but essential in the new role.

3. Product and Business Metrics
While QAs rely on KPIs such as test coverage, failure rate or bugs per release, the PO needs to monitor metrics such as retention, engagement, churn, LTV and NPS.
Changing the mindset from "product working properly" to "product generating results" is a game changer.

4. Decision Making in Ambiguous Environments
POs need to prioritize the backlog even when there is uncertainty, external pressures, and limited resources. The ability to make decisions based on trade-offs — and take calculated risks — doesn't usually come naturally to QAs, but it's indispensable for those leading product strategy.

**Skills a QA Must Develop to Become a PO**
The transition from QA to PO requires an intentional enablement effort.
Some essential skills include:

• Business Acumen
Understand how the product generates revenue, what the business model is, who the competitors are, and what the market forces are. Reading market reports, taking business analysis courses, and even a mini MBA are all effective ways to do this.

• Value-Based Prioritization
Master frameworks such as MoSCoW, RICE, WSJF and Kano Model to prioritize features based on business impact, technical effort and customer return.

• Storytelling and Assertive Communication
The PO needs to be able to sell ideas, engage the team and present results to different audiences. Developing soft skills such as public speaking, active listening and strategic writing is crucial.

• User Research and Hypothesis Validation
Participate in interviews, usability tests, heat map analysis and A/B testing. Knowledge of tools such as Hotjar, Google Analytics and Amplitude can be a differentiator.

• Product Metrics Management
Learn to work with activation, retention, CAC, LTV and conversion rate metrics. More than numbers, the PO must understand the "why" behind the data and use it to guide decisions.

Certifications that Drive Transition
Certifications are not prerequisites, but they add value and demonstrate commitment to the new role. Some recommended certifications include:

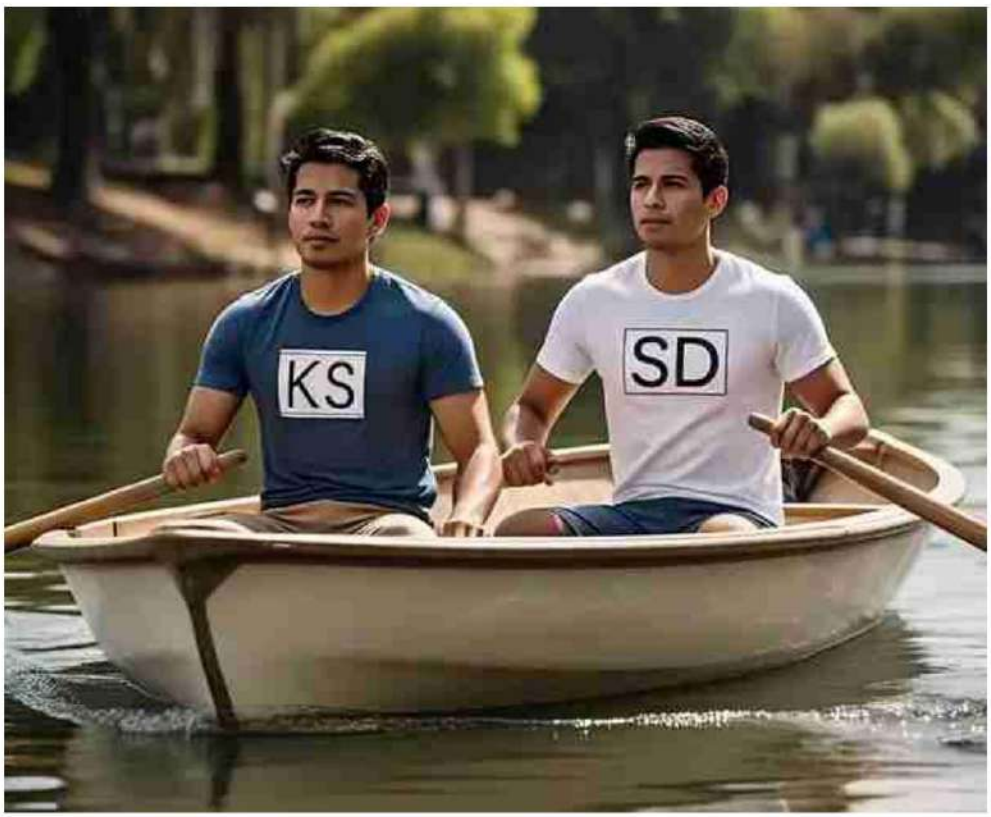
Certified Scrum Product Owner (CSPO)
Professional Scrum Product Owner (PSPO I and II) – Deepens product practices and metrics.

**Conclusion: Many** QAs are natural candidates for PO, but this transition must be done without rushing, acquiring the hard and soft skills necessary for the new role, without forcing the bar, in a conscious and purposeful way.

# Scrum for development and Kanban for support.

The software industry has evolved into a landscape where adaptability and continuous value delivery are essential.

In this context, **Scrum** and **Kanban** emerge not as rivals but as complementary approaches, each with its own strengths: the former for development projects, the latter for the unpredictable dynamics of support. This article explores how these frameworks coexist harmoniously, adapting to modern needs.




## 1-Evolved Scrum: Flexibility and continuous innovation

Scrum, traditionally associated with fixed sprints and deliveries at the end of each cycle, has undergone significant transformations. Today, it incorporates practices that make it more agile and adaptable:

-Continuous Delivery Decoupled from the Sprint Review
Adopting Continuous Deployment (CD) allows increments to be released at any time, as long as they meet the Definition of Done. This eliminates the exclusive reliance on Sprint Review for

deliveries, accelerating time-to-market without compromising the framework structure.

***The Review is used to review what was done during the Sprint, if were to deliver something it would be called Delivery meeting.***

**-Changes in the Sprint Backlog**
Although the Sprint Goal remains unchanged, the Sprint Backlog can be adjusted during the cycle, as long as it is negotiated with the Product Owner (PO) and does not compromise the main objective. For example, critical corrections or emergency demands can be incorporated, ensuring flexibility without destabilizing the planning (ok, ok. If your company has very little support, to the point of not harming the Sprint, Scrum is fine).

**-Accumulation of Papers with Responsibility**
In smaller or multidisciplinary teams, it is possible to accumulate functions (such as a Scrum Master also acting as a developer), as long as there is transparency and balance to avoid conflicts of interest.

This practice reflects the maturity of teams in agile environments.

**Adoption of Lean and Kanban Practices**
In recent years, Scrum has incorporated principles from Lean Kanban, such as WIP (Work in Progress) limitation, to reduce bottlenecks and improve workflow. This integration reinforces the convergence between the frameworks, as both share values such as transparency and continuous improvement.

# 2-Kanban in support: Simplicity and immediate response

While Scrum is suited to complex projects, Kanban shines in sustainment environments, where absolute unpredictability, little knowledge of end users, and a sense of urgency reign:

## Incidents in production do not wait for ceremonies.

Critical issues require immediate action, without waiting for planning or prioritization in sprints. Kanban, with its continuous flow and visual boards, allows teams to “pull” urgent tasks directly into execution, resolving issues in real time.

**Absence of ceremonies with specific dates**
Without fixed events like Daily Scrum or Sprint Review, Kanban offers a malleable structure, ideal for support teams dealing with

variable demands. Dynamic prioritization adjusts to the criticality of incidents, maintaining focus on resolution.

**A developer in two contexts**

There is no conflict in a professional simultaneously participating in a Scrum team (for development) and another Kanban team (for support), obviously with a drop in productivity.

The key is clarity of priorities:

In **Scrum,** the focus is on the Sprint Goal and incremental delivery;
In **Kanban,** urgency determines action, with flexibility for interruptions.

In software companies where people get caught up in last-minute additions to meetings, discipline, prescriptive corridor ceremonies (like Sprint Planning and Daily Scrum) enforce helping teams maintain momentum and focus.

The Scrum structure, with short sprints and clear goals, forces the decomposition of products into manageable steps, reducing the anxiety of "everything for yesterday" and promoting consistent deliveries.

## 3-It all depends on your shoe size.

When I mention support, I am referring to companies that carry out dozens, hundreds or even thousands of services per day, those that
serve end users with limited knowledge, for example.

For smaller-scale support, which can be negotiated directly with the Product Owner, Scrum does the trick.
Furthermore, Scrum does not prohibit working overtime; what really matters is using common sense.

**Conclusion:** Contextual Choice, Scrum and Kanban are complementary tools, not mutually exclusive. While Scrum revolutionized software development with its structuring and iterative cycles, Kanban offers agility for continuous operations. The evolution of Scrum, incorporating Lean practices, proves that adaptation is essential.

For Brazilian organizations, where acculturation is still a challenge, Scrum brings order and compass.

Kanban, with its simplicity, solves the chaos of support.
Both coexist under the agile umbrella, ensuring that innovation and operations move forward in sync.

# Scrum scaling frameworks comparison.

The adoption of agile methodologies and frameworks has become increasingly most common in technology companies, driving the delivery of value and customer satisfaction. However, as teams grow and projects become more complex, the need arises of frameworks that support scalability. In this context, frameworks como Nexus, Large Scale Scrum (LeSS) e Scaled Agile Framework (SAFe) have gained popularity. Here, we will explore the characteristics of each and discuss why SQLive chose by Nexus.

## Nexus

Nexus is a framework developed by Scrum.org to connect multiple Scrum teams, allowing them to work in sync on a single Product. It maintains the essence of Scrum, adding only the elements necessary for integration between teams.

**Nexus Features:**

-Simple and Minimalist: Nexus does not create new roles beyond those defined in Scrum, maintaining the traditional Scrum team structure.

-Product Focus: Nexus is focused on the Product and the value delivered to the client, ensuring that all teams work towards the same goal.

-Team Integration: The Nexus defines three additional artifacts: the Nexus Product Backlog, the Nexus Increment, and the Nexus Integration Plan.

-Ceremonies: Nexus adapts Scrum ceremonies to accommodate scale, including Nexus Planning, Nexus Daily Scrum, Nexus Review, and Nexus Retrospective.

Nexus Promises:

-Facilitate the transition for teams already familiar with Scrum.

-Ensure the implementation of consistent Scrum practices across all teams.

-Encourage collaboration and integration between teams.

# Large Scale Scrum (LeSS)

LeSS is a framework that applies Scrum principles on a large scale, with as little change as possible. It is based on the idea that many of the problems faced by growing agile teams

can be resolved by more rigorously applying Scrum principles.

LeSS Features:

-Focus on Scrum Framework: LeSS maintains the traditional Scrum framework, with Scrum teams working on a single Product Backlog.

LeSS Principles:
- The framework defines principles such as "Feature Teams", "Component Teams are Bad" and "One Product Backlog".

-Ceremony Adaptation: LeSS adapts Scrum ceremonies to accommodate scale, including LeSS Planning and LeSS Review.

LeSS Promises:

-Promote collaboration and shared responsibility between teams.
-Ensure that all teams work towards the same goal.
-Encourage continuous improvement and adaptation.

# Scaled Agile Framework (SAFe)

SAFe is a comprehensive framework for large-scale agile adoption. It provides a detailed structure for managing multiple agile teams, including development teams, product management, and other functions.

SAFe Features:

-Level Structure: SAFe defines four levels: Team, Program, Grand Solution and Portfolio.

-Ceremonies and Practices: SAFe includes ceremonies such as PI Planning, Daily Scrum and System Demo.

-Focus on Architecture: SAFe emphasizes the importance of architecture and the technical solution.

SAFe Promises:

-Provide a comprehensive framework for managing multiple agile teams.

-Ensure alignment between teams and strategic objectives.

-Foster collaboration and value delivery.

All three frameworks - Nexus, LeSS and SAFe - aim to scale your agile practices. However, I prefer Nexus because of its simplicity and strict adherence to Scrum principles, without any new roles or paraphernalia.

## Conclusion

Choosing a scalable agile framework is strategic and directly impacts the effectiveness and performance of teams. LeSS and SAFe have their advantages. merits, but SQLive found the ideal solution in Nexus: simplicity and total alignment with Scrum. With this decision, the company scaled agility without bureaucracy, maintaining the essence of Scrum without introducing unnecessary functions or steps.

I honestly have reservations about frameworks that try to make Scrum 'more palatable to those coming from waterfall', and for those who believe in complex control structures it is definitely not the Scrum.org approach. , and this

# The fantastic certificate factory.

What do I think about all these frameworks and certifications outside of Scrum.org and Scrum Alliance?

There are two certainties in life: death, the release of new JavaScript frameworks, and Scrum certifications.

Most of them set out to fix what isn't broken, make Scrum more palatable to non-agility believers, and make a lot of money.

I personally support Scrum.org as I find it more rigorous and fair.

Scrum.org charges for certifications, requiring 85% accuracy, and not for courses in which the certificate comes out practically in a gumball machine.

Recertify someone after two years when there have been no changes in Scrum guide, it is almost offensive.

By any chance, when they launched the window heater, did they ask all drivers to get their driver's license again?

## Conclusion

Don't waste your money on certifications of dubious quality, they won't will bring practical results to your career.

# The Scrum Team and AI:

# Opportunities and Challenges.

Artificial Intelligence (AI) is revolutionizing the way software development teams operate, and the Scrum framework is also affected by this change. By automating routine tasks,
predictive analytics and decision optimization, AI has the potential to increase team productivity while also presenting new challenges for agile teams. We will discuss how AI influences each role within Scrum (Scrum Master, Product Owner, and
Developers), the challenges to be , and how this technology can applied to improve efficiency.

## 1. Impact of AI on Scrum Roles

### a) Scrum Master

The Scrum Master's mission is to eliminate obstacles and ensure that the team follows agile principles. Artificial intelligence can help with this task in different ways.

**-Ceremony Automation:** AI tools can schedule meetings, generate automatic Daily Standups summaries, and identify communication patterns that can be improved.

**-Retrospective Analysis:** Platforms like **Fireflies.ai** or **Otter.ai** can transcribe and analyze discussions, identifying recurring themes and suggesting improvements.

**-Risk Detection:** Algorithms can predict potential delays based on data and metrics such as WIP, Throughput, Cycle time and capacity.]

**-AIs can create dynamic scripts:** Based on dynamics found in books and articles, they do not have the creativity to create new dynamics.

**Challenge:** The Scrum Master must learn to understand the data produced by artificial intelligence and ensure that recommendations are in line with agile principles, avoiding becoming too dependent on automation.

***Try one day to generate an article by an AI and you will see the amount of wrong data and outdated views: Opinions like: You can only deliver in the review, use velocity instead of throughput to calculate Wip, and other nonsense.***

## b) Product Owner (PO)

The PO is responsible for the backlog and prioritization of stories. AI can bring the following benefits:

**-Intelligent Prioritization:** Tools like **Aha!** or **Jira AI** can suggest the order of the backlog based on market data, user feedback, and development cost.

**-User Story Generation:** The different language models can help you write clearer descriptions.

**-Feedback Analysis:** AI can process large volumes of feedback from customers (reviews, support) to prioritize improvements.

**Challenge:** The PO will need to validate the AI's suggestions, keeping the focus on the real value for the user and not just on automated metrics (AI analyzes data and trends, it has no feeling or creativity).

## c) Development Team

For developers, QA´S (a developer specialized in testing), designers (a developer specialized in UI), AI can speed up repetitive tasks and improve quality.

**-Code Generation:** Tools like **GitHub Copilot** and **Amazon CodeWhisperer** suggest code snippets, reducing development time.

**-Automated Testing:** AI can create and run tests based on code changes, identifying bugs faster.

**-DevOps Optimization:** Systems like **Datadog** and **New Relic** use AI to detect anomalies in real time.

**Challenge:** The team must thoroughly scrutinize the code produced by AI to prevent security and quality issues. Furthermore, reliance on tools can reduce the overall understanding of the system.

I've seen a lot of mediocre SQL code generated by AIs, for example.

## 2. New Challenges Introduced by AI in Scrum

**-Ethics and Bias:** AI-based decisions may reflect biases.

**-Loss of Human Touch:** Excessive automation can reduce team collaboration and creativity.

**-Maintain Agile Focus:** Remember that AI is a productivity tool, nothing more than that, and you will be held accountable for any errors.

# MY OPINION(2025)

In my view, there is currently an overvaluation of technologies.
artificial intelligence that currently exists. Unfortunately, the AI industry and certain unethical consultants are propagating the idea that in the near future there will be no need for developers at all.

I predict that the reality will be harsh, resulting in many implementation errors, vulnerable code, rigid interfaces, as well as financial losses and disappointments.

Artificial intelligence will be constantly advancing and at an accelerated rate.
Currently, their presence is essential, although a large part of their functions consists of accessing information available to humans.

(like on Stack Overflow, for example) and adapt codes accordingly
the requests received.

It does what developers have done for decades, but it still struggles with fine-tuning.

I have seen IAs suggesting mediocre solutions in my area of expertise (SQL).

There is no such thing as a so-so APP, an ugly and inflexible APP will not be used, this applies to every item in a system's architecture.
There is no somewhat secure system either.

And today AIs are far from the quality of good programmers.

**According to Mark Zuckerberg:**
**“In 18 months coding will be done by AI, it will be better than the work of most engineers. ”**

For me it is a fallacy, both in terms of time and the type of coding that AIs will be able to create in the medium term. We are in June 2025, they charge me in December 2026, and in June 2036.

Quality is fundamental in Scrum and can be improved over time, but it must always be based on solid foundations, never starting from a mediocre base, or one that is not 100% reliable.
In short, AI is a useful tool, but it is still far from 100% replacing the developer.

Don't be part of the majority mentioned by the Meta owner, be a reference.
Don't fight AI, be the best at using it.

# Conclusion

Artificial Intelligence has the ability to transform the way Scrum teams operate, providing increased productivity but also presenting new challenges. The secret lies in balancing automation and critical thinking, ensuring that

technology acts as a facilitating instrument, and not as a substitute for human creativity and cooperation. Teams that know how to incorporate Artificial Intelligence strategically will be in a better position to offer more innovative and higher added value products.

# The Evolution of Agile Frameworks and the Need for Adaptation in the Microservices Era

## *Agile Frameworks and the Architecture of the 90s*

Most agile frameworks, including Scrum, emerged in the 1990s, a period when software development was still largely shaped by monolithic architectures. In this scenario, a product increment was typically associated with a single implementation of new features in a system that contained a large range of functionality. Delivery followed a more linear flow, and the Sprint Review served as a focal checkpoint for validation before release into production.

## *Microservices and the fragmentation of increment*

With the advent of microservices, the dynamics have changed. A single team Scrum often works on multiple microservices simultaneously, which means that, in the same sprint, several independent increments may be ready for production at different times. In this scenario, waiting for the *Review* to release a microservice that has already been tested and validated may not make sense – especially if it does not have critical dependencies with other development items.

## *The Importance of Dependency Management in the Sprint Backlog*

When teams work with microservices, the *Sprint Backlog* becomes more complex due to the interdependencies between tasks. If a microservice depends on an API that is being developed in parallel, releasing it before that API is completed can lead to inconsistencies. Therefore, it is essential to map and control these relationships to avoid:

-Partial deployments that break functionality

-Rework due to poorly planned integrations

-Delays caused by bottlenecks in critical dependencies

## *Visualizing Dependencies: Graphs and Tools*

To help with this management, some maps and graphics are extremely useful:

### Dependency Graph

-Shows the relationships between *Sprint Backlog* items in node format and edges.

-Identifies microservices that can be released early (without blocking) and those that need to wait for other components.

### Dependency Matrix

-A table that cross-references services and indicates which ones have dependency relationships (for example, "Service A needs Service B's endpoint").

### Deployment Roadmap

-A visual schedule that defines the ideal release order based on interdependencies.

## *Conclusion*

Traditional agile frameworks need to be adapted to microservices scenarios, where independent incremental deliveries are common. The Sprint Review maintains its value as a ceremony to review what has been done and to realign, but teams should have the autonomy to release to production when it makes sense – as long as dependencies are clearly mapped. Visual tools, such as
graphs and matrices are essential to avoid risks and ensure that agility is not lost amid modern architectural complexity.

# No, it wasn't all bush!

A lie that has been spread by word of mouth in recent years is that before the agile manifesto, only Waterfall was used.

In my professional life, I have never needed to make a flowchart.

In fact, when I worked for banks in the early 90s I used waterfall, and the worst thing is that it worked, since the banks were not startups that were inventing something revolutionary, they knew what they wanted, most of the time we just developed the same software that the competition had, there were no surprises, agile methodologies help to tame uncertainty.

When I set up my Software House, at the end of 1993, I started exploring market niches such as private security and mail companies, health plans, advertising agencies, foundation construction companies, and continuous form factories, companies that at the time had no system at all.

I had to deal with clients (I didn't even know the term stakeholder), I financed the development and owned the copyright, that is, I didn't suffer any kind of pressure due to deadlines, the competition was small, when I thought the product was ready for the market
used the salespeople at the continuous form factory to sell them.

It was very easy to get started, until the software orders arrived, and I had to start hiring programmers, as Zagalo would say: “Then **we were surprised again!”**

It was very difficult to scale my “DOITNOWBRO” methodology, not that I didn’t try.

My programmers had to use my code, my standardization, I tried to supervise the quality, pass guidelines, inspect what was done, but it didn't always work.

Some were good “mini Pos/developers”, some were not.
Because the developer was sometimes too nice and accepted every little idea suggested, even if it didn't add any value to the product, others misunderstood what the client wanted, others didn't understand that "packaged software" had to work for several companies and not for the one he was developing at the time.

Needless to say, my social life ended during this period, working at least 15 hours a day.

I tried to invent the “Daily”, I tried to catch people by surprise when they got close to the coffee machine, and I asked if anyone had any difficulties, I shared experiences “from the front”, perhaps that was the secret to the relative success I had at the time.

But it was almost impossible to get the programmers together, some were on the phone, others were rushing to finish jobs.

Methodologies like Scrum didn't just put an end to Waterfall, they brought peace and discipline to those who were already averse to it as well.

The pillars and values of Scrum helped developers understand the value of quality and continuous improvement.

A Daily, a short alignment, always at the same time, with the same duration, in the same place.
No one makes any other commitments for the time, and they already know what they have to say, they know they have to be direct.

User Stories replaced the old functional specifications of the Waterfall folks, but through the Product Backlog, they increased transparency for anyone using something like my "DOITNOWBRO" as well.

Review ensures that things don't get out of control, both for those who delivered in waterfall mode and for those who used the agile "methodologies" of the time.

Anyone who criticizes Daily's TimeBox has missed the point.

The Planning, Review and Retrospective TimeBoxes are your maximum duration times, often taking less than 30% of the time.

In other words, the time spent on the "bureaucratic part" is infinitely less than the time spent on the useless Waterfall junk, and it does not burden at all those who used "Stone Age SCRUMS".

In larger software houses there has always been a figure who did not program, such as the Scrum Master, but he was a figure who was concerned with demands and not with the dissemination of values that lead to quality and process improvements.

Recently, a person related to Caixa Econômica Federal said that Scrum Masters are useless, almost useless were the old report, spreadsheet and schedule fillers from fantasy island.

Being a Scrum Master in a state-owned company in Brazil must not be easy.

Remembering that: in small companies there may be an accumulation of roles, a programmer can also play the role of Scrum Master.

## *Conclusion*

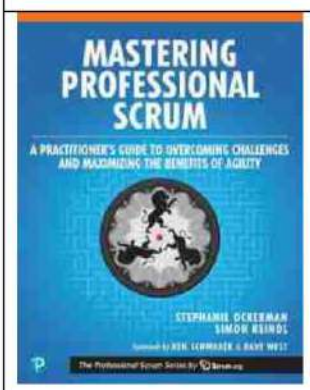
Agility has always existed, what almost didn't exist was methodology.

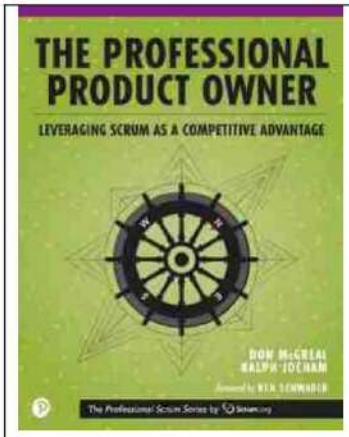
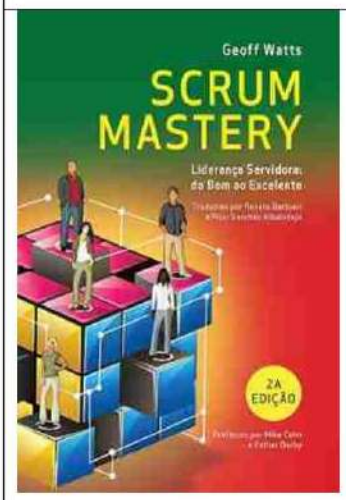
# Here are some books I recommend about Scrum.

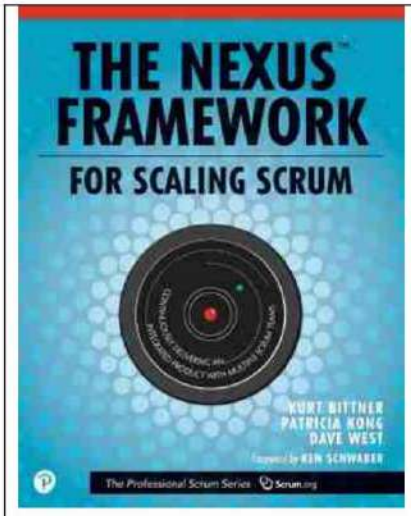
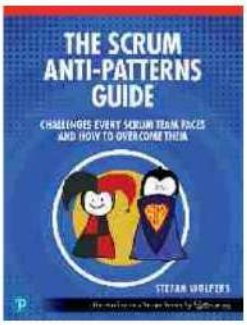
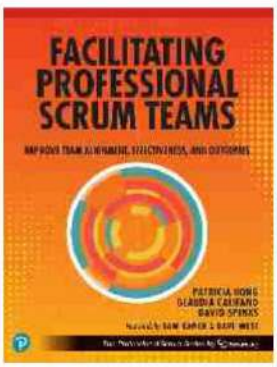
Most of the books currently available are not up to date with the current Scrum Guide (2020). Below, I present my perspective on books that I believe would add to the knowledge of

who wants to work with Scrum, in its various roles.

They are all in English, if you want to work with Scrum, you will need to master the language, or get by with a translator.

| | |
|---|---|
|  | **Applying Scrum with Kanban: A Pointless Book! -2021-Andy Hiles**<br><br>Book that covers the techniques of Scrum with Kanban shows how storypoints are inefficient and shows in a way fun as assembling and visualizing the workflow. Leading and lagging indicators, definition of workflow, etc. |
|  | **Mastering Professional Scrum: A Practitioner's Guide to Overcoming Challenges and Maximizing the Benefits of Agility (The Professional Scrum Series) (English Edition)**<br><br>Everyday tips on removal of impediments, clear examples of definition of done, How to build a great Scrum team. |

| | |
|---|---|
|  | **The Professional Product Owner: Leveraging Scrum as a Competitive Advantage Paperback – Illustrated, June 4, 2018**<br><br>English Edition by Don McGreal (Author), Talks about the responsibilities of the product owner within the Scrum framework (how he should act in each ceremony, about the types of PO and how to improve yourself as a professional, hard skills and soft skills. It shows that the Po is a value maximizer and not a bureaucratic project manager. |
|  | **SCRUM MASTERY**<br>**Servant Leadership: from Good to Excellent-Geoff**<br>**-Watts-2nd edition-**<br>**Portuguese**<br>Strategies to Make Scrum Meetings More Productive<br><br>The qualities and skills of a great Scrum Master<br>Techniques to improve team performance<br>Tips for strengthening team engagement<br>Ways to encourage the team<br>how to encourage collaboration and creativity |

| | |
|---|---|
|  | **The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams Capa comum – Ilustrado, 17 dezembro 2017**<br><br>-Describes in detail the implementation of the Nexus Framework-A framework for scaling Scrum. How to manage the dependencies of multiple scrum teams working under the command of a single PO and a single backlog, from the definition<br><br>of done shared until the Nexus Integration Team (NIT) is up and running. |
|  | The Scrum Anti-Patterns Guide<br>The book covers the most common anti-patterns that appear in projects that use Scrum. Come in they have problems in communication, difficulties in preparing the (Definition of<br><br>Done), lack of transparency and respect by the principles of Scrum. |
|  | Facilitating Professional Scrum Teams<br>Facilitating alignment, facilitating the Planning of Sprint, Making Daily Easy Scrum, facilitating the dynamics of team.<br>Techniques for conducting meetings productive daily routines, whether for status updates or to solve problems quickly.<br>Strategies for dealing with challenges such as conflicts and improve decision making |

If this booklet was useful to you, please consider donating any amount (R$ 30?) to Some of the associations that help autistic children in Brazil. Autism today has grown 3000% compared to decades ago. The causes are still being studied, but the needs of children cannot wait. Autistic children lack not only psychiatric help, but also nutritional and educational help.

To donate via PIX to an association for autistic children, you can search for the PIX key of the institution you wish to support, such as AUMA (Association of Friends of Autistic Children) or ABADS (Brazilian Association for Social Assistance and Development). AUMA uses the PIX key 38.893.038/0001-03 and ABADS uses the key doe@abads.org.br.

# About the Author:

Marcello Eduardo , began his studies in IT IN 1986 in the laboratories of Colégio Comercial Brasil in Vila Carrão-SP, he has a postgraduate degree in Software Engineering from Bráz Cubas.
Has the following certifications: PSM1, PSMPO1, PSK1 and SPS from Scrum.org

During his career he worked for companies such as Mais Soft, Aon, Fidelity, Brasfond, SQLive and Ultralub.

https://www.linkedin.com/in/marcello-eduardo-de-oliveira-dias-721201362/