Introduction to The Developers

Exploring Developer accountability in Scrum across various domains.

by Mayko Silva

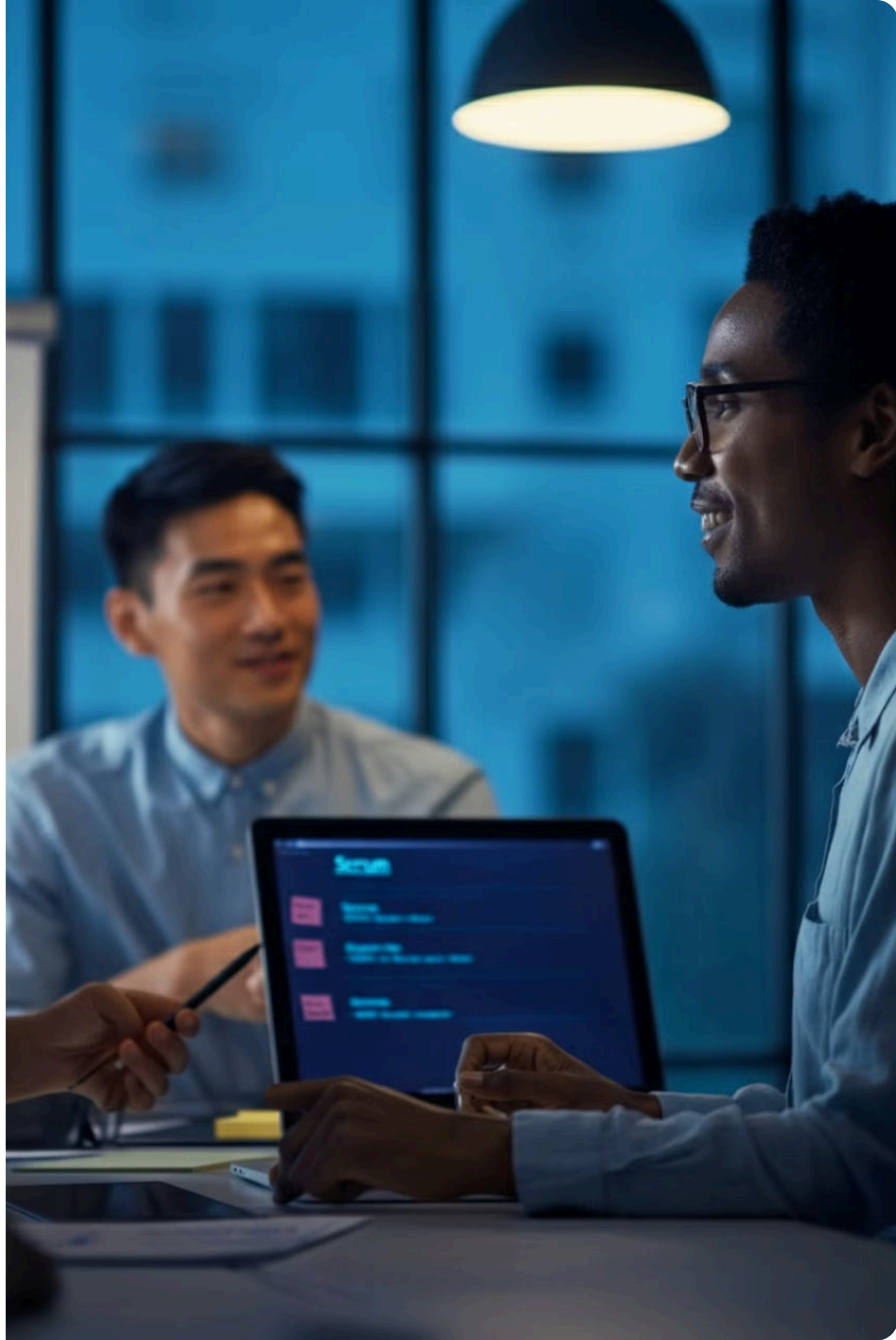
Module Overview

1 **Developer Accountability**

Exploring Developer accountability in Scrum

2 **Broader Concept**

Beyond traditional "developer" role



Topics We'll Cover

Developer Identity

Who is a Developer?

Development Focus

Feature Development vs. Overall Accountability

Core Responsibility

The Developer's Job: Create

Developer Accountabilities

The Developer's Job: We'll delve into the specific accountabilities that Developers hold within the Scrum framework, including planning, quality assurance, adaptation, and professional accountability.

Topics We'll Cover

Instilling Quality

We'll look at how Developers ensure the quality of their work through the Definition of Done.

Daily Adaptation

We'll explore how Developers adapt their work on a daily basis to meet the Sprint Goal.

Interpersonal Accountability

We'll discuss how Developers hold each other accountable as professionals.

What's not said about Developers

We'll uncover some aspects of the Developer's role that aren't explicitly stated in the Scrum Guide.

Topics We'll Cover

Adding New Developers

We'll look at when and how new Developers should be added to a Scrum Team.

The Impact of Adding Developers

Finally, we'll discuss how adding new Developers can affect team dynamics and productivity.

Challenging Misconceptions

Beyond Software

"Developers" in Scrum aren't just software developers

Inclusive Role

Include anyone contributing to creating the Increment

Cross-Domain Application

Applicable across domains: construction, manufacturing, engineering, etc.

Practical Scenarios We'll Explore

1

Task Allocation

How do Developers decide task allocation?

2

Goal Alignment

Ensuring alignment towards the same goal

3

Adaptability

Handling unexpected situations

Module Objectives

1 **Role Understanding**

Developer's role in Scrum

2 **Accountability Awareness**

Their accountabilities

3 **Team Dynamics**

How they work within the team

4 **Value Creation**

Their contribution to creating valuable Increments

Remember

Beyond Software

Think beyond software development

Versatility

Scrum is versatile and applicable to various fields

Key Understanding

Understanding this versatility is key to grasping Developer's role

Ready to Dive In?

Explore

Let's explore the world of Developers in Scrum!

Who is a Developer?

by Mayko Silva

Key Quote from Scrum Guide

> "Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint."

- Developers are **committed team members** in Scrum
- They focus on **creating usable Increments**
- Their work spans **any aspect** of the Increment
- This commitment is ongoing, happening each Sprint

Beyond Software Development

Scrum's Broad Applicability

Scrum is not limited to the software field

Inclusive Definition of Developer

Developer: anyone contributing to Increment creation

Diverse Domains

Construction

Developer might be painting walls

Deserted Island

Developer could be digging irrigation trenches

Beekeeping

Developer might manage hives or process honey

Focus on Value Creation

Actively Working Towards Value

Developers are actively working towards creating a usable and valuable Increment

Contribution Over Titles

It's not about job titles, but about the contribution to the product

Cross-functional Skills

Collective Skills

Developers possess a range of collective skills to create value each Sprint

- Coding
- Testing
- Design
- Writing

Value Creation

These diverse skills enable developers to contribute effectively to the project's success and deliver valuable outcomes within each Sprint

Practical Example: Eco-friendly House

Architect

Designing layout for eco-friendly house

Solar Panel Expert

Installing systems for renewable energy

Landscaper

Planning water-efficient gardens

Interior Designer

Focusing on sustainable materials for interior

Scrum Team Developers:

- Architect designing layout
- Solar panel expert installing systems
- Landscaper planning water-efficient gardens
- Interior designer focusing on sustainable materials

Understanding Check

In a Scrum Team developing a new restaurant concept, which of the following would be considered Developers?

A) Only the software engineers working on the ordering system

B) The chefs creating the menu, the interior designers, and the software engineers

C) Only the chefs creating the menu

D) Only the restaurant manager

Answer and Explanation

The correct answer is B. This example illustrates how developers in various fields can contribute to creating a usable Increment. In this scenario, chefs, interior designers, and software engineers are all actively involved in developing different aspects of the restaurant concept.

Chefs are developing the menu and culinary experience, interior designers are creating the ambiance and physical space, and software engineers are likely working on systems for reservations, ordering, or kitchen management. Each of these professionals is a "developer" in their own right, contributing their expertise to create a valuable and functional end product - in this case, a fully operational restaurant.

This example reinforces the idea that development extends beyond just software creation. It showcases how cross-functional teams with diverse skills can work together to create value, which is a core principle in modern development practices like Scrum.

Remember

Focus on Contribution

Focus on contribution to Increment, not job titles

Value Creation

Consider all aspects of value creation

Beyond Software

Think beyond software - Scrum applies in various fields

Key Takeaways

Broad Definition

Broad definition of Developers crucial for effective Scrum implementation

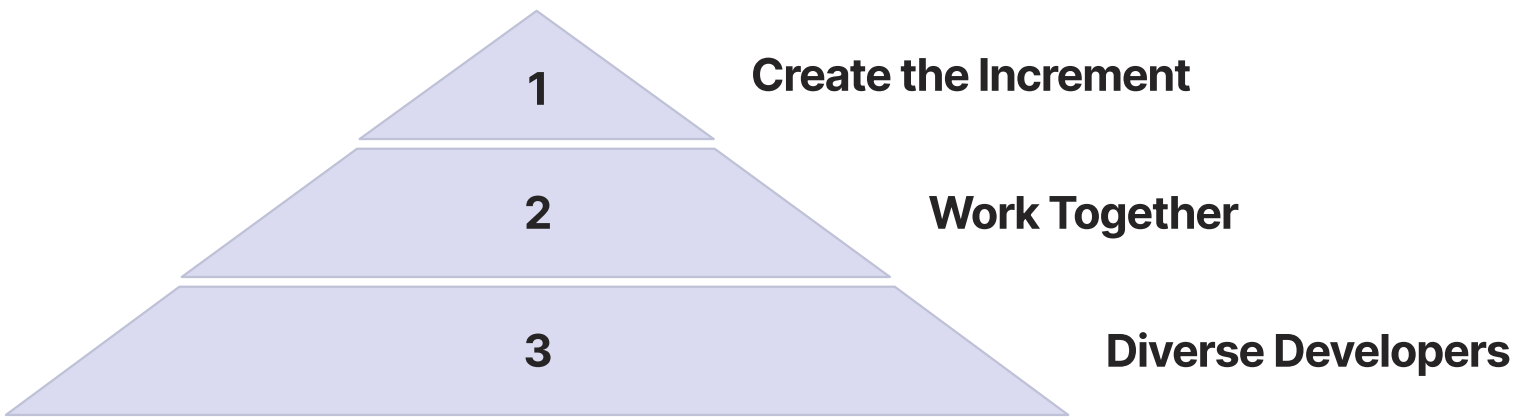
Collaboration

Promotes cross-functionality, collaboration, holistic value creation

Value Creation

In Scrum, if you're actively contributing to creating value, you're a Developer!

Next Steps



Exploring how diverse Developers work together to create the Increment

Feature Development vs. Overall Accountability

by Mayko Silva

The Nuanced Difference

Key Question 1

Who is accountable for creating a useful Increment?

Key Question 2

Who is committed to creating aspects of a usable Increment?

Scrum Guide Statements

The Scrum Guide provides clear statements regarding team accountability and developer commitment in the Scrum framework. These statements highlight the distinct roles and responsibilities within a Scrum Team:

"The entire Scrum Team is accountable for creating a valuable, useful Increment every Sprint."

This statement emphasizes the collective responsibility of the entire team in delivering value. It underscores the importance of collaboration and shared accountability in achieving sprint goals.

"Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint."

This second statement focuses on the specific role of developers within the Scrum Team. It highlights their commitment to actively creating the components that make up the usable increment in each sprint.

Understanding the Difference

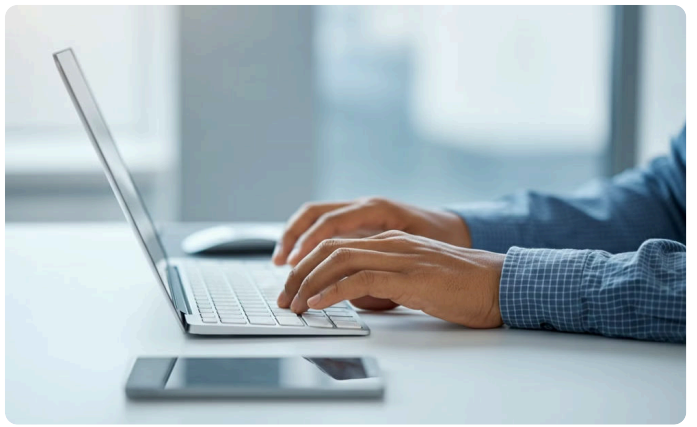
Team Accountability

Whole Scrum Team responsible for overall value and usefulness of Increment

Developer Commitment

Developers commit to creating individual aspects or features

The Big Picture



Developers' Focus

Developers focus on "how" of creating features

Team's Focus

Entire team focuses on "why" and "what" of creating value

Collaboration

Promotes collaboration and shared responsibility

Practical Implications

Collaborative Development

Developers not working in isolation

Team-Oriented Value Delivery

Part of team effort to deliver value

Specialized Roles

Product Owner and Scrum Master contribute through specific accountabilities

Practical Example: Mobile Banking App

Developers

Coding new features

Product Owner

Aligning features with needs

Scrum Master

Facilitating processes

Entire Team

Accountable for value

In our mobile banking app example, we can see how different roles contribute to the overall accountability:

- Developers focus on coding new features like bill payment and account transfers
- The Product Owner ensures these features align with user needs and market demands
- The Scrum Master facilitates processes and removes impediments
- The entire team is accountable for delivering an app update that provides real value to users

This example illustrates how individual responsibilities contribute to the collective accountability for the project's success.

Understanding Check

In a Scrum Team developing a new electric car, who is accountable for ensuring the car meets safety standards?

A) Only the Developers working on safety features
B) Only the Product Owner
C) Only the Scrum Master
D) The entire Scrum Team

Answer and Explanation

Correct answer: D

- **Entire Scrum Team accountable** for car meeting safety standards
- **Developers** implement safety features
- **Product Owner** ensures alignment with regulations and market expectations
- **Scrum Master** helps team work effectively towards goal

Remember

Team Accountability

Entire Scrum Team accountable for overall value and usefulness of Increment

Developer Commitment

Developers committed to creating specific aspects or features

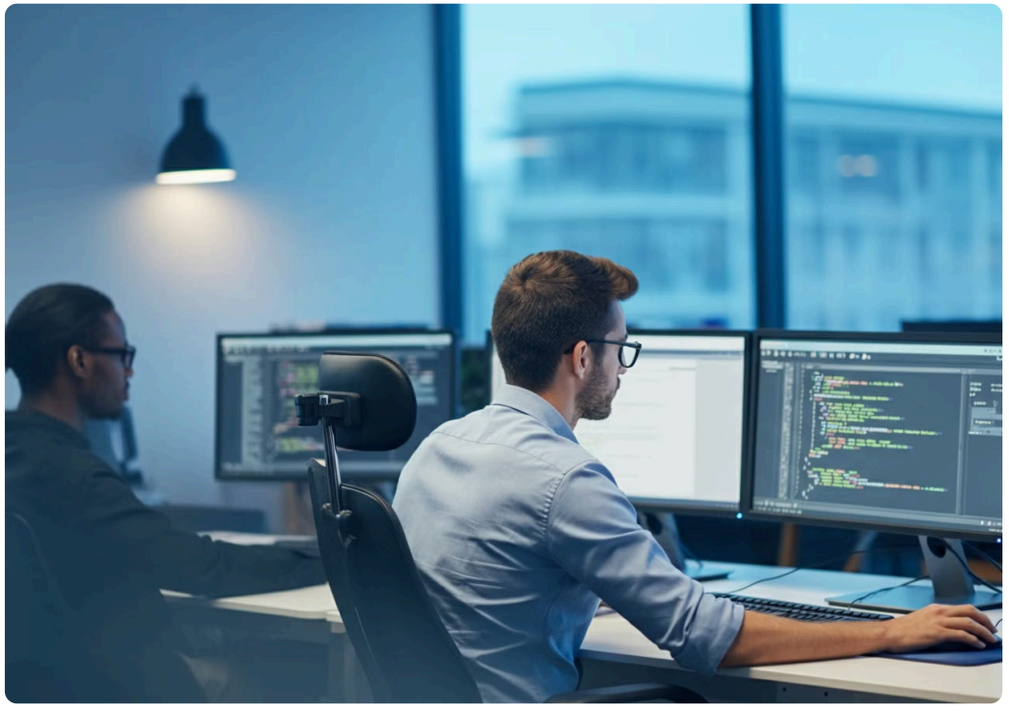
Collaboration

Distinction promotes collaboration while recognizing specific contributions

Key Takeaway

Developer Focus

Developers concentrate on creating features



Team Focus

Entire team remains focused on delivering value

Understanding this nuance is crucial for effective Scrum implementation:

- Developers focus on creating features
- Entire team remains focused on delivering value

Next Steps

1

Explore Daily Practices

Investigate how shared accountability manifests in day-to-day Scrum activities

2

Analyze Team Dynamics

Observe how team members collaborate and support each other in Scrum events

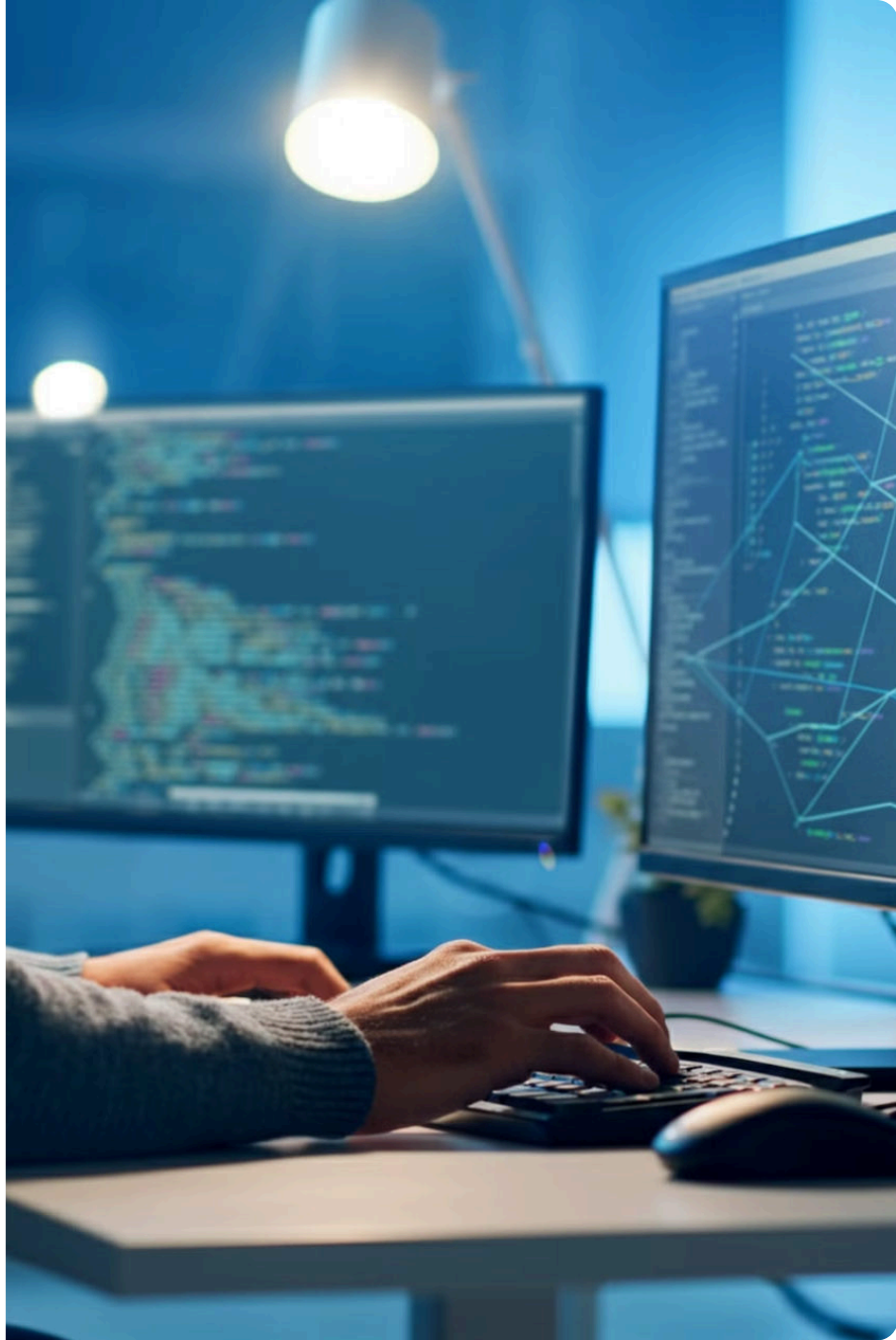
3

Evaluate Outcomes

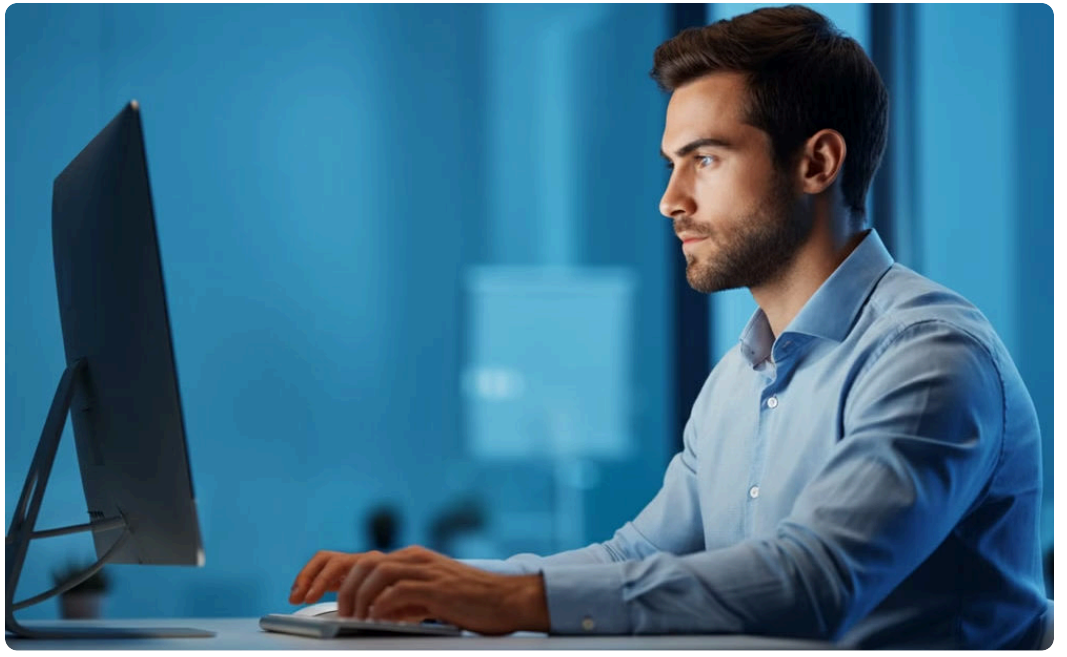
Assess the impact of shared accountability on project success and team morale

The Developer's Job is Simple: Create

by Mayko Silva



The Simplicity of Creation

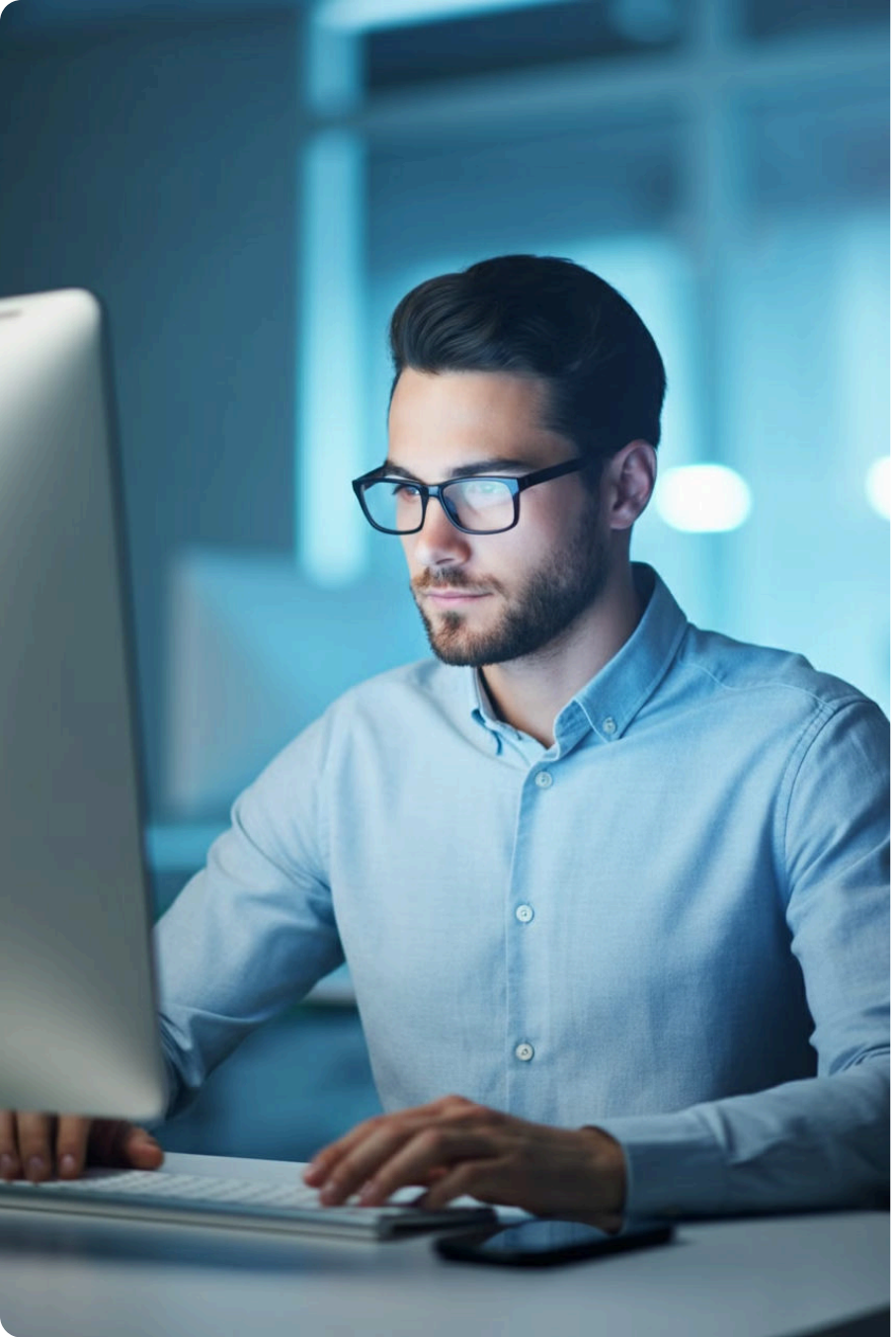


Core Job of Developer

The core job of a Developer in Scrum is to create

Scrum Guide Reflection

This simplicity is reflected in the brief Developer section in the Scrum Guide



Focus on Creating Value

- Simple core job allows focus on creating value
- Eliminates distractions and unnecessary complexities

The Four Key Accountabilities

1 **Creating a plan for the Sprint**

Developers are responsible for creating the Sprint Backlog

2 **Instilling quality**

Adhering to the Definition of Done to ensure quality

3 **Adapting plan each day**

Continuously adjusting the plan toward achieving the Sprint Goal

4 **Holding each other accountable**

Maintaining professional accountability within the team

Unpacking the Accountabilities

1

Planning

Active participation in Sprint Planning

2

Quality

Adhering to Definition of Done

3

Adaptation

Daily adjustments to align with Sprint Goal

4

Accountability

Self-managing team responsible for performance

The Balance of Simplicity and Responsibility

The developer's job, while simple in concept, requires a high level of skill to execute effectively. This balance of simplicity and responsibility allows developers to focus on the crucial aspects of their work without getting bogged down in unnecessary complexity.

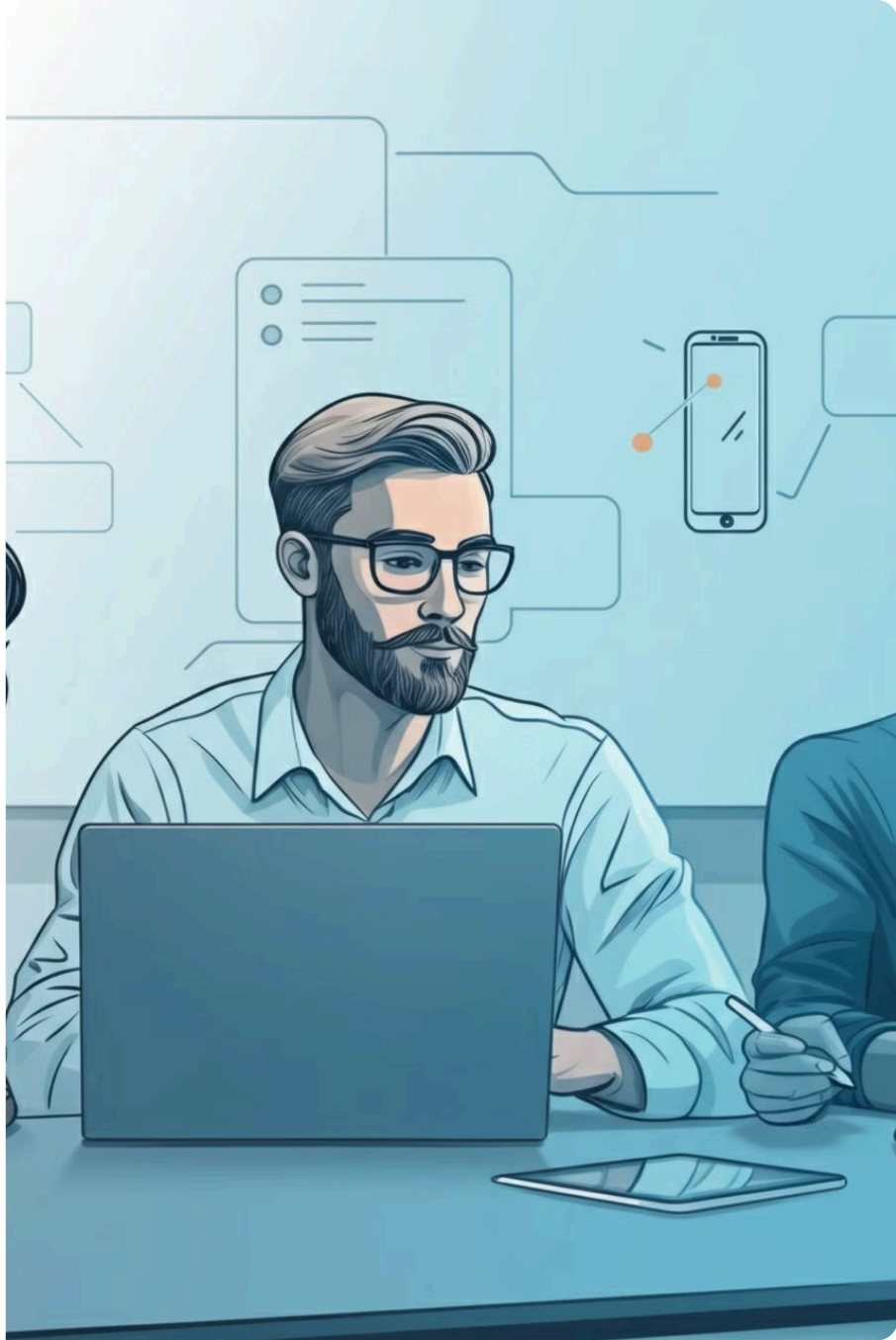
By maintaining this balance, developers can concentrate their efforts on creating value and meeting their key accountabilities. This approach enables them to navigate the complexities of software development while keeping their primary objectives clear and achievable.

Practical Example: Mobile App Development

In the context of mobile app development, the developers' primary job is to create the app. This seemingly simple task encompasses a range of responsibilities:

- Planning features for the app
- Ensuring quality throughout the development process
- Adjusting plans as needed
- Working effectively as a team

These aspects demonstrate how the core job of creation involves multiple interconnected activities, all contributing to the successful development of the mobile application.



Understanding Check

Which of the following best describes the Developer's job in Scrum?

A) To follow detailed instructions provided by the Scrum Master

B) To create value by developing aspects of a usable Increment each Sprint

C) To manage the Product Backlog and stakeholder relationships

D) To oversee the work of other team members

Answer and Explanation

Correct answer: **B**

- Developer's job: Create value by developing aspects of usable Increment each Sprint
- Encapsulates simplicity of core job (create)
- Acknowledges focus on value and iterative nature of Scrum

Key Reminders

Simple Core Job

Developer's core job is simple: create

Focus on What Matters

Simplicity allows focus on what truly matters

Skill and Collaboration

Important accountabilities require skill and collaboration

Balance for Success

Balance of simplicity and responsibility enables focus and flexibility

Key Takeaway

Balance is Crucial

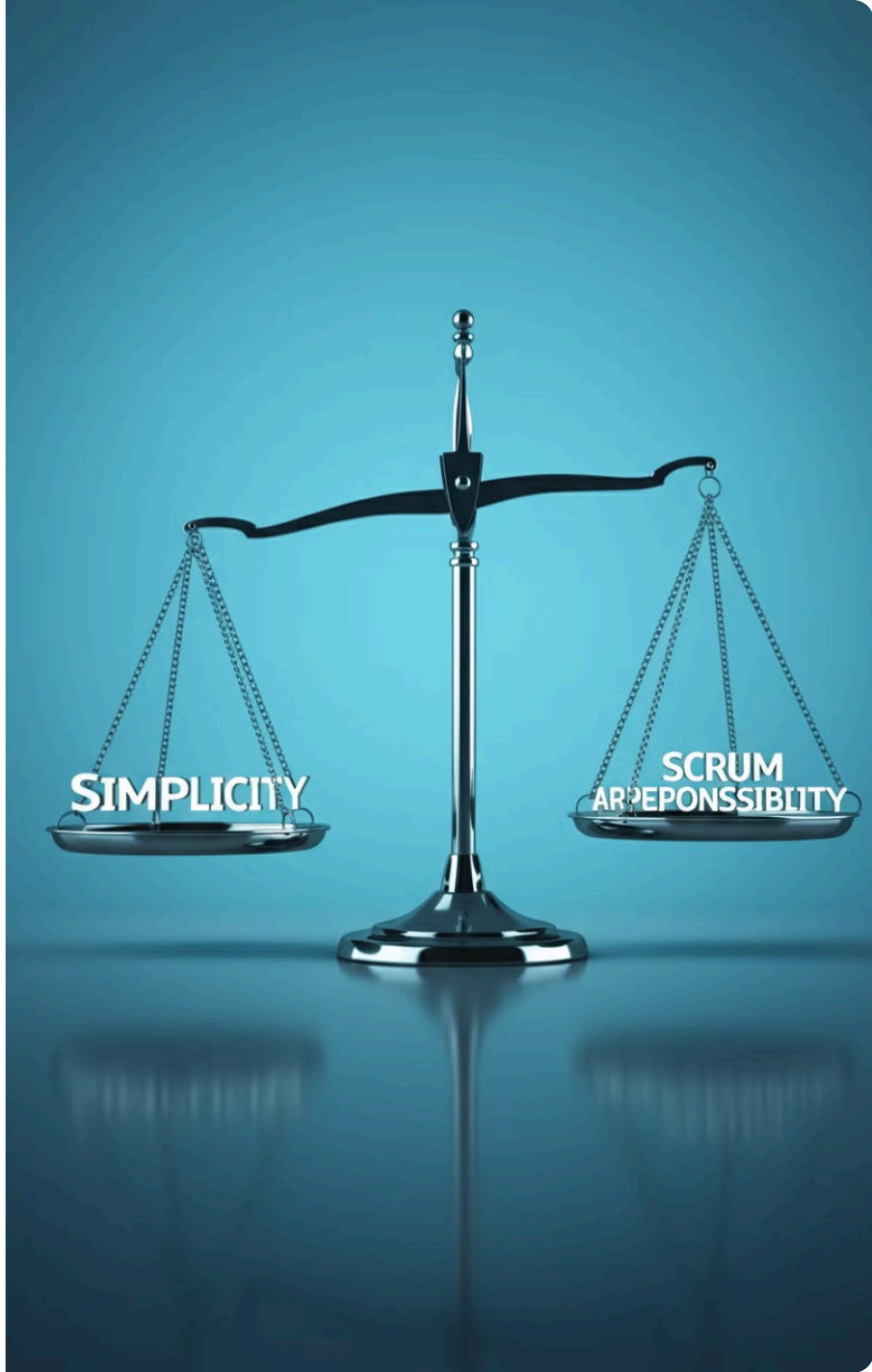
Understanding the balance between simplicity and responsibility is crucial for effective Scrum implementation

Concentrate on Value

Allows concentration on creating value

Maintain Adaptability

Maintains adaptability at heart of Scrum



Next Steps

Dive Deeper

Developer Accountabilities

Scrum Framework

Diving deeper into how Developers fulfill accountabilities within Scrum framework

Developer Accountabilities

Key responsibilities and expectations for software developers.

by Mayko Silva

Creating the Sprint Backlog

Accountable for Sprint Plan

Developers create the Sprint Backlog

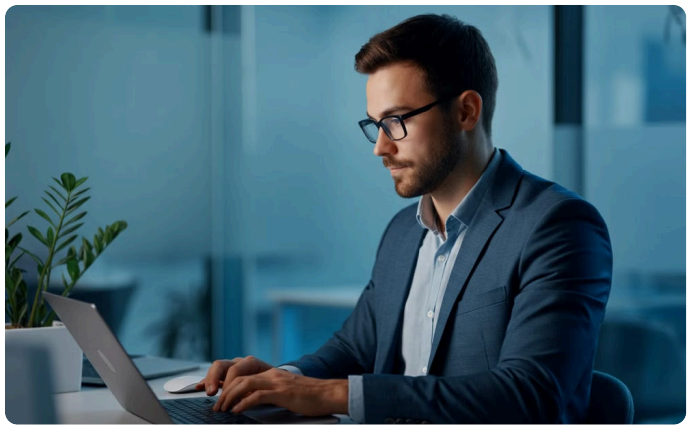
Select from Product Backlog

Choose items to accomplish in Sprint

Plan Sprint Execution

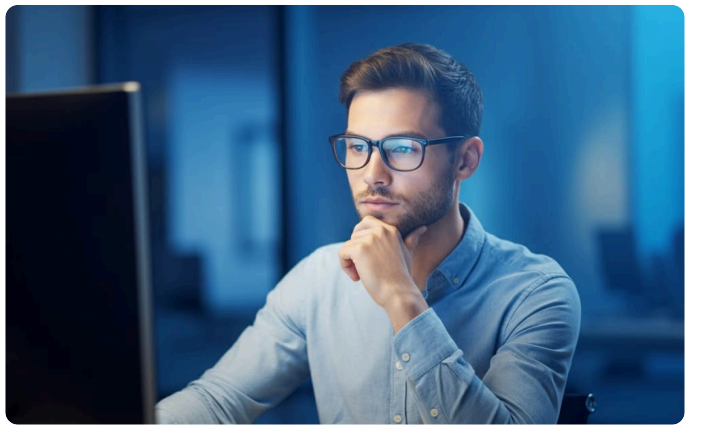
Determine how to complete Sprint items

Instilling Quality



Adhering to Definition of Done

Responsible for meeting team standards



Ensuring Increment Quality

Verifying each deliverable meets team standards

Quality from the Start

Building quality into the development process

Daily Adaptation

Adapt Plan

Adjust plan daily toward Sprint Goal

Inspect Progress

Review progress during Daily Scrum

Flexibility

Respond to changes to meet Sprint Goal

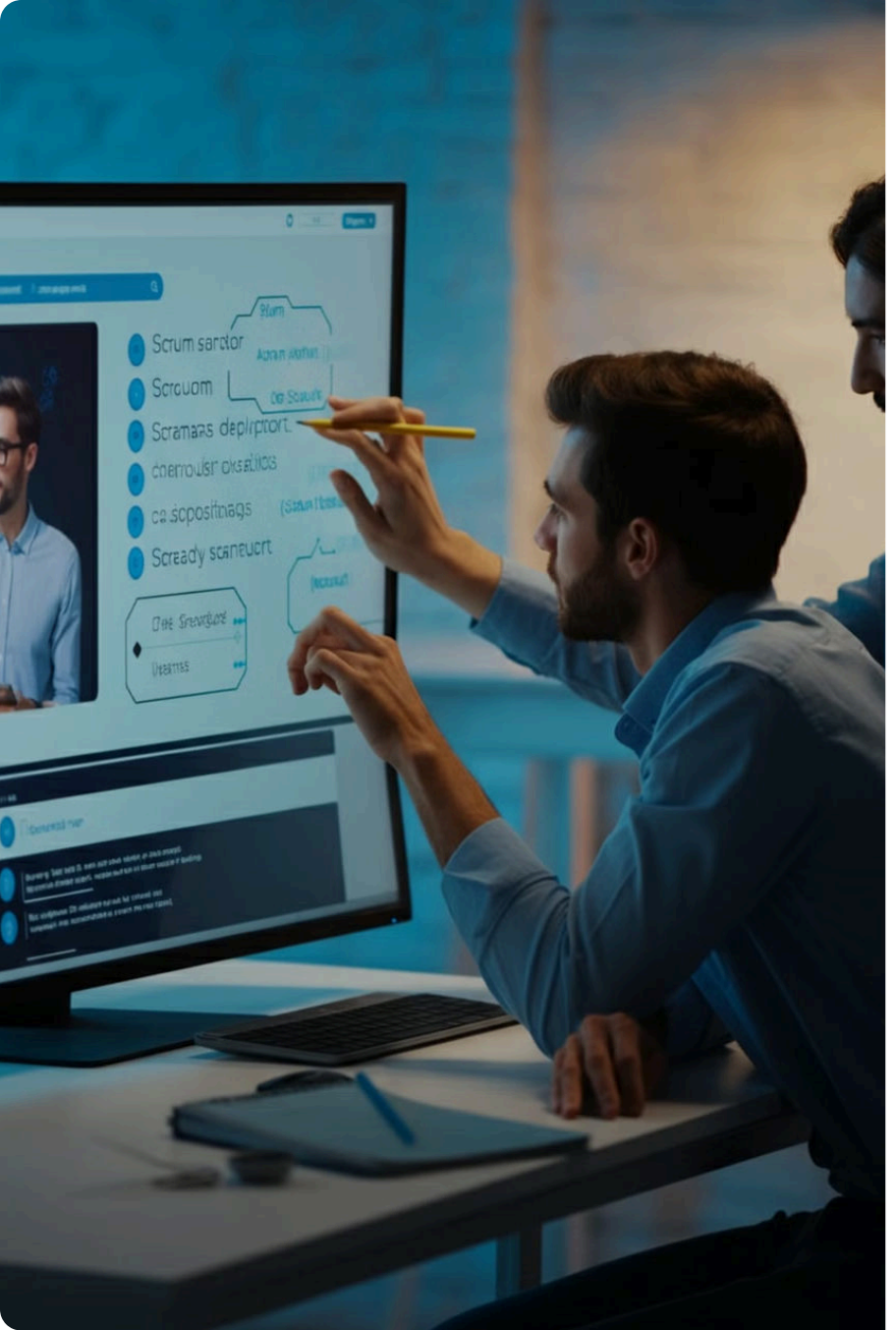
Professional Accountability

Hold Each Other Accountable

Foster a culture of shared responsibility

Support Continuous Improvement

Not about blame, but delivering best results



Ownership of the Sprint Backlog

Developers Manage Backlog

Only developers can add or remove items from the Sprint Backlog

Empowers Self-Management

Allows the team to effectively manage their own work

Key Aspect of Scrum

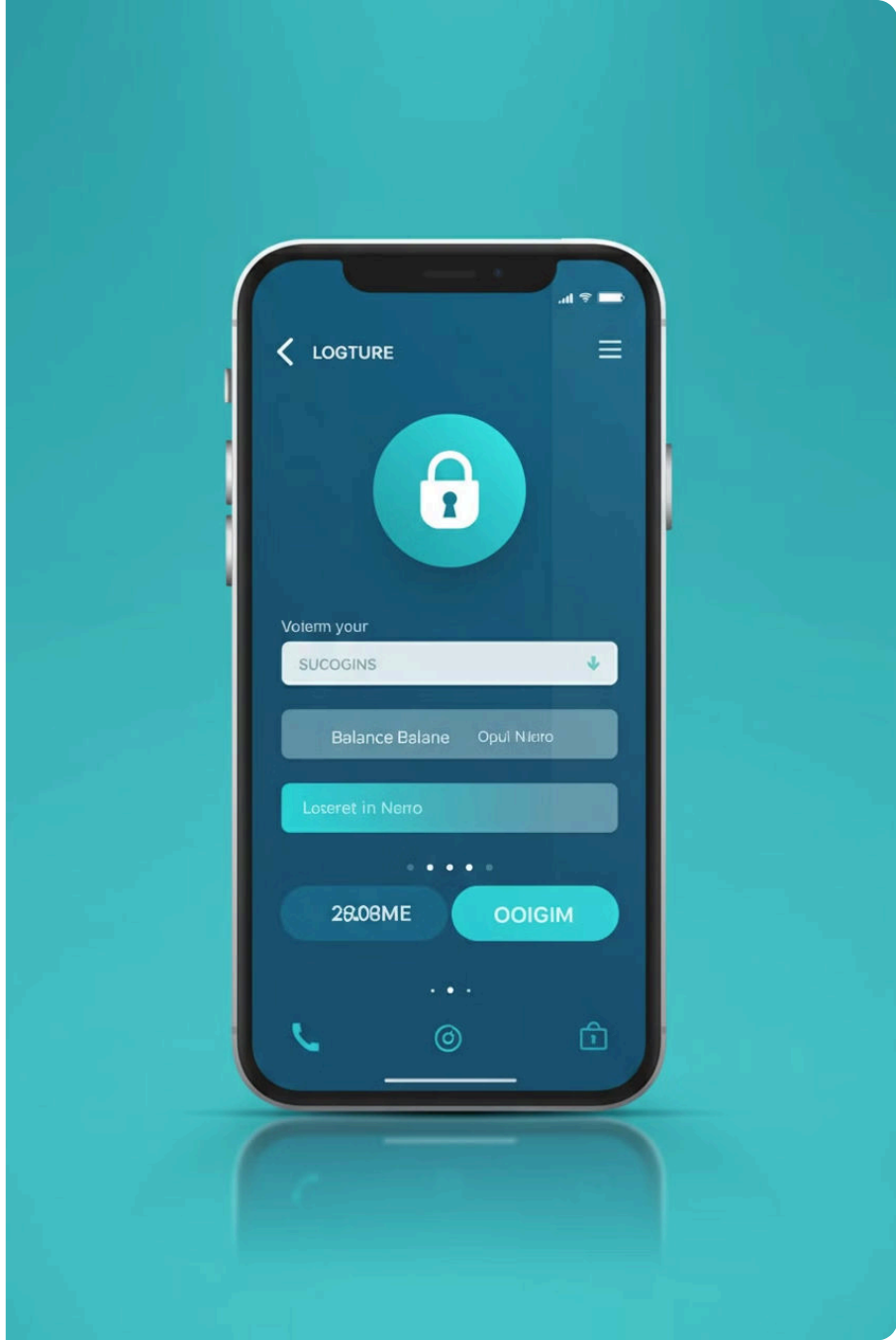
Self-management is a core principle of the Scrum framework

Practical Example: Mobile Banking App

Create Sprint Backlog for secure login and balance checking features. Adhere to Definition of Done with security testing and user acceptance.

Adapt daily based on progress and challenges. Hold each other accountable for coding standards and pair programming.

Add tasks like optimizing database queries without external approval.



Understanding Check

Who has the authority to add new items to the Sprint Backlog during a Sprint?

A) The Scrum Master

B) The Product Owner

C) The Stakeholders

D) The Developers

Answer and Explanation

Only Developers Add/Remove Items

Developers have authority over Sprint Backlog

Self-Management and Ownership

Crucial aspect of Agile development

Correct Answer: D

Developers control the Sprint Backlog

Remember

Developers Create Backlog

Developers own and create the Sprint Backlog.

Quality Through Definition

Quality is instilled through adherence to Definition of Done.

Daily Alignment

Daily adaptation aligns the team with the Sprint Goal.

Professional Accountability

Developers hold each other professionally accountable.

Key Takeaway

1. **Empowers Developers**

   Developers take ownership of their work.

2. **Ensures Alignment**

   Aligns with Scrum Team and product goals.

Next Steps

1

Explore Accountabilities

Understand how Scrum accountabilities play out in day-to-day practices

2

Continuous Improvement

Identify opportunities to enhance Scrum implementation and team performance

3

Practical Application

Apply learnings to real-world Scrum scenarios and projects

Instilling Quality

by Mayko Silva

The Definition of Done

Scrum Guide Definition

"Developers are always accountable for instilling quality by adhering to a Definition of Done."

Purpose

Agreed-upon set of items for product backlog item to be "done"

Function

Serves as quality gate for all increments

Quality as a Continuous Process

- Quality built into development process from start 🏗️
- Developers maintain high standards throughout Sprint 🚀

Role of Definition of Done

Creates Transparency

Establishes a shared understanding of what "complete" means, fostering clarity and alignment within the team.

Prevents "99% Done" Syndrome

Eliminates ambiguity and ensures tasks are truly finished, avoiding the common pitfall of nearly-complete work.

Ensures Team-Wide Understanding

Provides a clear definition of "done" for product backlog items, promoting consistency across the entire team.

Creating Definition of Done

1

Collaborative Creation

The Definition of Done is collaboratively created by the Scrum Team, ensuring all perspectives are considered.

2

Continuous Evolution

It evolves over time as the team learns and improves, adapting to new insights and challenges.

3

Potential Inclusions

- Code reviews
- Testing
- Documentation
- Performance criteria

Quality and the Scrum Pillars

Transparency

Supports transparency by clarifying quality criteria

Inspection

Enables inspection with clear evaluation standard

Adaptation

Facilitates adaptation by highlighting improvement areas

Practical Example: Mobile Banking App

Code Quality

- Peer-reviewed code
- Passing unit and integration tests

Compatibility

- Multi-device and OS testing

Documentation

- Updated user documentation

Security

- Completed security audit

Performance

- Met performance benchmarks

Understanding Check

Let's test your understanding of quality assurance in Scrum with a multiple-choice question:

What is the primary mechanism for ensuring quality in Scrum?

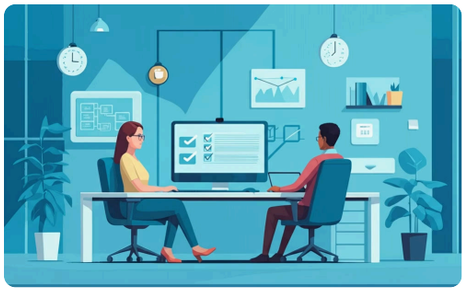
- A) The Scrum Master's oversight
- B) The Product Owner's approval
- C) The Definition of Done
- D) The Product Goal

Answer and Explanation

The correct answer is: **C**

- Definition of Done is the primary quality assurance mechanism
- Serves as a quality gate for all increments

Key Reminders



Quality through Definition of Done

Quality instilled by adhering to Definition of Done

Transparency and Shared Understanding

Definition of Done creates transparency and shared quality understanding

Continuous Quality Process

Quality is continuous, not added at end

Evolution of Definition of Done

Definition of Done evolves with team learning

Supporting Scrum Pillars

Supports Scrum pillars: transparency, inspection, adaptation

Key Takeaway

Empowering Developers

Understanding Definition of Done's role is crucial for effective Scrum as it empowers Developers to own quality

Ensuring Valuable Increments

Definition of Done ensures delivery of valuable, high-quality increments each Sprint

Understanding Definition of Done's role is crucial for effective Scrum:

- Empowers Developers to own quality
- Ensures delivery of valuable, high-quality increments each Sprint

Next Steps

Exploring how Developers adapt work daily to meet Sprint Goal

Daily Adaptation

by Mayko Silva

The Principle of Daily Adaptation

- **Scrum Guide:** "Developers are always accountable for adapting their plan each day toward the Sprint Goal."
- Requires flexibility and responsiveness to changes and new information

The Sprint Backlog as a Living Document

The Sprint Backlog serves as the Developers' plan for the Sprint, providing a crucial roadmap for the team's work. However, it's important to understand that this document is not set in stone. Instead, the Sprint Backlog is designed to evolve throughout the Sprint, adapting to new insights, challenges, and opportunities that arise during the development process.

This flexibility allows the development team to respond effectively to changing circumstances and emerging requirements. By treating the Sprint Backlog as a living document, teams can ensure that their work remains aligned with the Sprint Goal while accommodating necessary adjustments along the way.

The dynamic nature of the Sprint Backlog reflects the Agile principle of embracing change and continuous improvement. It empowers the Developers to make informed decisions about their work, optimizing their approach as they gain more information and experience throughout the Sprint.

Continuous Inspection and Adaptation

In Scrum, the process of continuous inspection and adaptation is a fundamental principle that drives project success. This approach involves the constant inspection of progress throughout the development cycle. As teams work through their sprints, they are continuously evaluating their progress, identifying obstacles, and assessing the effectiveness of their current strategies.

Based on these actual experiences and observations, teams adapt their plans accordingly. This flexibility allows for real-time adjustments to be made, ensuring that the project remains on track and aligned with the evolving needs of stakeholders. The ability to adapt plans based on empirical evidence is a key strength of the Scrum framework.

This practice of continuous inspection and adaptation aligns perfectly with Scrum's empirical approach to project management. By basing decisions on observed facts rather than predictions or assumptions, teams can make informed choices that lead to better outcomes. This empirical process control helps teams navigate the complexities and uncertainties inherent in software development and other complex projects.

The Role of the Daily Scrum

1

15-Minute Timebox

Daily Scrum is a brief, focused meeting for Developers

2

Inspect Progress

Developers review progress toward Sprint Goal

3

Adapt Sprint Backlog

Make necessary adjustments to the Sprint Backlog

4

Plan Next 24 Hours

Outline work to be done in the coming day

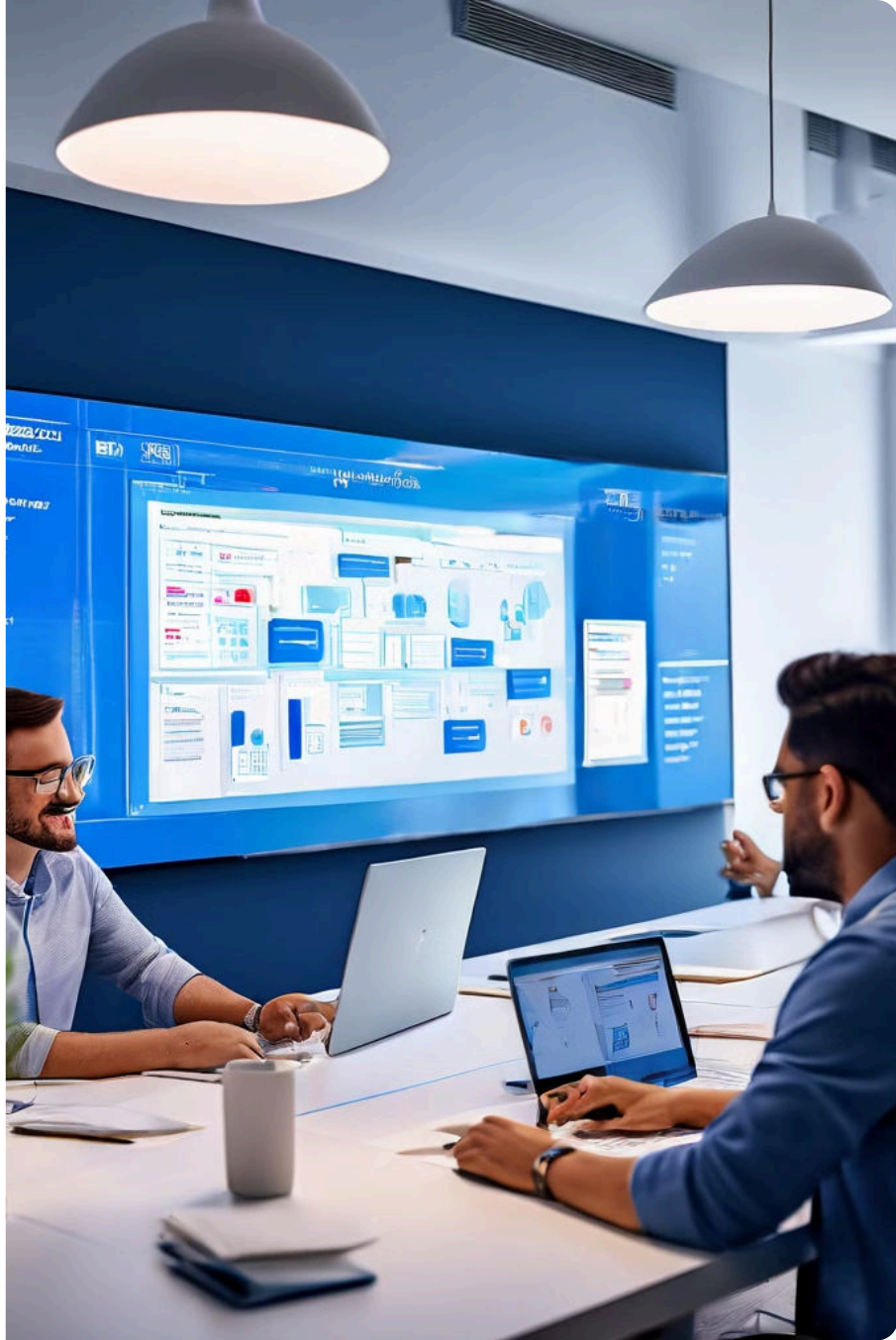
Flexibility Beyond the Daily Scrum

Anytime Adaptation

Adaptation encouraged anytime during Sprint if necessary

Sprint Backlog Modifications

Can involve adding, removing, or modifying Sprint Backlog tasks



Balancing Adaptation with Commitment

- Changes should serve achieving Sprint Goal
- Not about deviating from Sprint Goal

Practical Example: Mobile Banking App

Let's explore a practical example of daily adaptation in a Scrum project for a mobile banking app. The Sprint Goal is to implement a secure login feature.

Day 1: Plan

Fingerprint authentication tasks

Day 3: Adapt

Incorporate facial recognition

Day 7: Optimize

Code performance improvement

Day 10: Enhance

Add security testing tasks

This example demonstrates how the team adapts their Sprint Backlog throughout the Sprint:

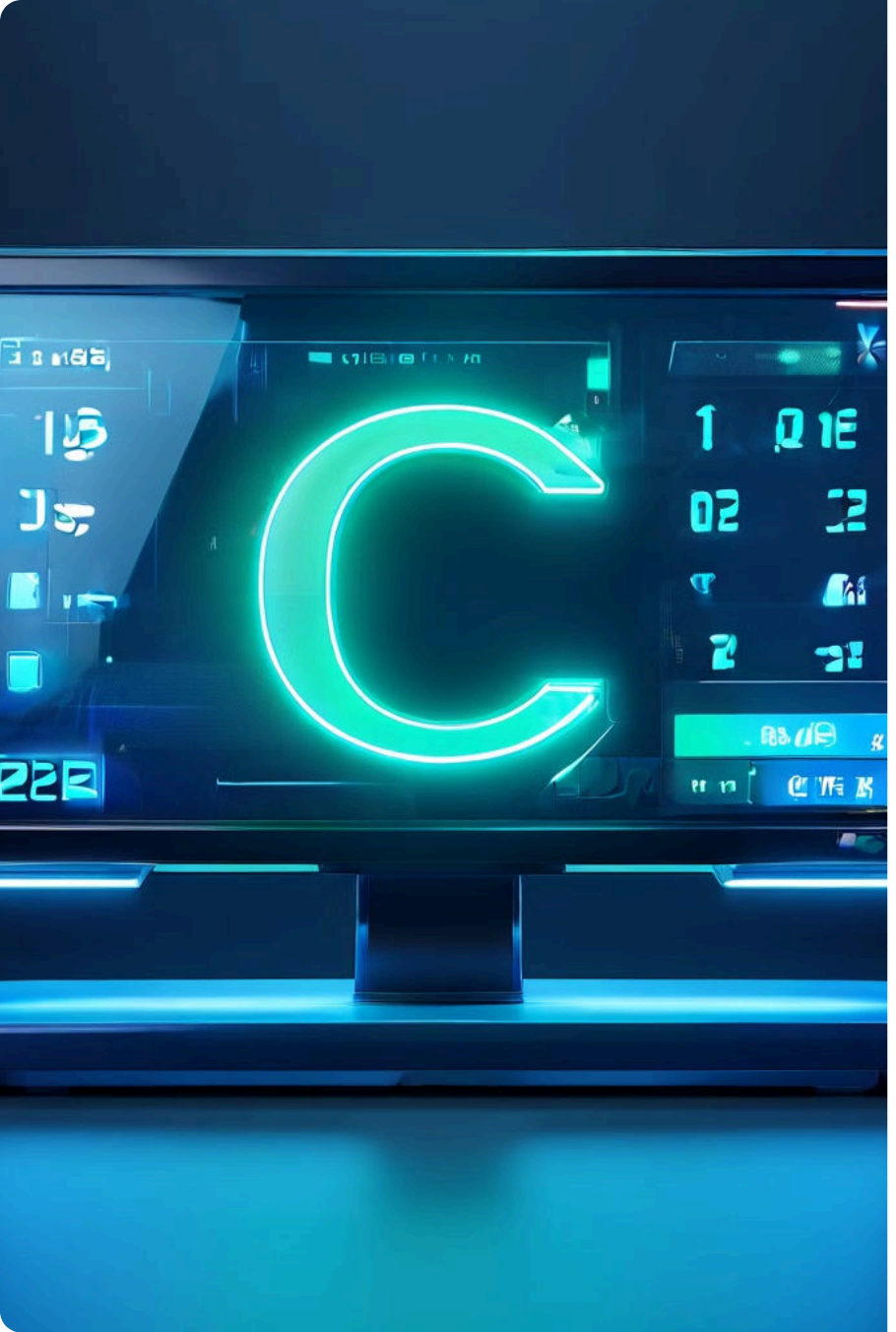
- Day 1: They plan fingerprint authentication tasks
- Day 3: They adapt to incorporate new facial recognition technology
- Day 7: They adapt again to optimize code for performance
- Day 10: They add security testing tasks to the Sprint Backlog

This practical example illustrates how a Scrum team can remain flexible and responsive to new information and changing priorities while working towards their Sprint Goal of implementing a secure login feature.

Understanding Check

When are Developers allowed to adapt their plan by changing the Sprint Backlog?

A) Only during Sprint Planning
B) Only during the Daily Scrum
C) Any time during the Sprint
D) Only during the Sprint Review



Answer and Explanation

Correct answer: C

- Developers can adapt plan by changing Sprint Backlog any time during Sprint
- Daily Scrum is formal opportunity, but not only time for adaptation

Remember

Daily Adaptation and Accountability

Daily adaptation is key Developer accountability

Evolving Sprint Backlog

Sprint Backlog evolves throughout Sprint

Adaptation for Sprint Goal

Adaptation serves Sprint Goal achievement

Daily Scrum Importance

Daily Scrum crucial for adaptation, but not only time

Balancing Act

Balance flexibility with commitment

Key Takeaway

Response to New Information

Daily adaptation allows teams to respond effectively to new information as it arises during the Sprint.

Overcoming Obstacles

Implementing daily adaptation helps teams identify and overcome obstacles quickly, maintaining Sprint momentum.

Continuous Improvement

Daily adaptation enables continuous improvement in achieving the Sprint Goal, enhancing overall Scrum effectiveness.

Next Steps

Exploring how Developers hold each other accountable as professionals

Interpersonal Accountability

by Mayko Silva

The Principle of Interpersonal Accountability

- **Scrum Guide:** "Developers are always accountable for holding each other accountable as professionals."
- Responsible for own work and team's work as a whole

Self-Management in Action

Key Component

Self-management is a key component of Scrum Teams

Empowerment

Empowers Developers to own work and team performance

Reduced Oversight

Reduces reliance on external oversight

The Role of the Scrum Master

In the Scrum framework, the Scrum Master plays a crucial role, but it's important to understand that this role does not involve holding Developers directly accountable. Instead, the Scrum Master acts as a coach and facilitator, rather than a traditional manager or supervisor.

The primary responsibility of the Scrum Master is to support the team in practicing accountability. This means creating an environment where team members feel empowered to hold themselves and each other accountable, rather than relying on external enforcement.

By focusing on coaching and facilitation, the Scrum Master helps the team develop the skills and mindset necessary for effective self-management. This approach fosters a culture of shared responsibility and mutual support, which is essential for the success of Scrum teams.

Daily Opportunities for Accountability

Daily Scrum

Regular accountability opportunity

Progress Discussion

Developers share updates

Plan Sharing

Team outlines next steps

Obstacle Identification

Addressing challenges

The Daily Scrum provides a regular opportunity for accountability within the team. During this meeting, developers discuss their progress, share their plans, and identify any obstacles they're facing. This structure enables peer-to-peer accountability, allowing team members to hold each other responsible for their commitments and progress.

Beyond the Daily Scrum

- Accountability is ongoing throughout Sprint
- Encourages open communication, help-offering, issue-addressing

Balancing Accountability and Support

Focus on Quality

Not about blame or criticism

Focuses on supporting high-quality work delivery

Collaborative Approach

Involves constructive feedback, knowledge sharing, collaborative problem-solving

Practical Example: Mobile Banking App

In this real-world scenario, we see interpersonal accountability in action within a development team working on a mobile banking app:

- Developer A demonstrates accountability by pairing with Developer B, who is struggling with their tasks. This collaborative approach helps address challenges and improve overall team performance.
- Developer C shows transparency and accountability by openly discussing their difficulties in keeping up with the workload. The team responds by redistributing tasks, ensuring a balanced and efficient workflow.
- When a quality issue arises, the entire team takes collective responsibility. They work together to improve processes, showcasing a commitment to continuous improvement and shared accountability for the project's success.

Understanding Check

Who is primarily responsible for holding Developers accountable for their work in Scrum?

- A) The Scrum Master
- B) The Product Owner
- C) The Developers themselves
- D) The stakeholders

Answer and Explanation

The correct answer is: **C**

- Developers themselves are primarily responsible for mutual accountability
- This is a key aspect of self-management in Scrum Teams

Remember

Core Developer Responsibility

Interpersonal accountability is core Developer responsibility

Mutual Accountability

About mutual accountability, not external oversight

Scrum Master's Role

Scrum Master facilitates but doesn't enforce

Balanced Approach

Accountability paired with support and collaboration

Continuous Process

Daily Scrum key opportunity, but accountability is continuous

Key Takeaway

Empowerment

Understanding and implementing interpersonal accountability is crucial for effective Scrum. It empowers Developers to own their work and team performance.

Culture Building

Interpersonal accountability fosters a culture of responsibility and continuous improvement within the Scrum team.

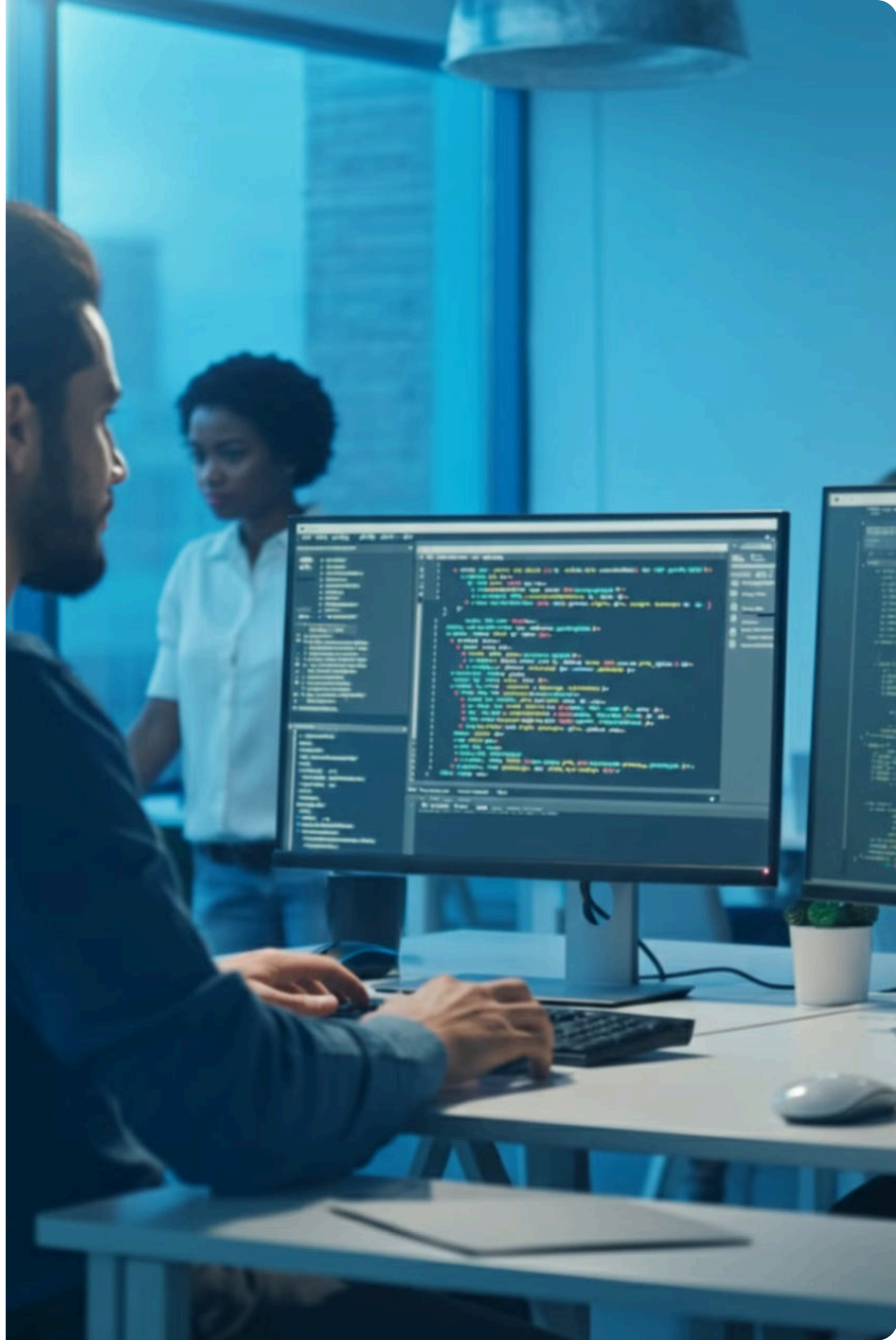
Next Steps

Our journey through the intricacies of interpersonal accountability in Scrum doesn't end here. As we continue to explore and deepen our understanding of Scrum practices, it's crucial to look beyond what's explicitly stated in the Scrum Guide. Our next area of focus will be examining what the Scrum Guide doesn't say about Developers and how these omissions impact Scrum practice in real-world scenarios.

This exploration will provide valuable insights into the nuances of the Developer role and how teams can leverage these unspoken aspects to enhance their Scrum implementation. By delving into these areas, we'll uncover new opportunities for improving team dynamics, productivity, and overall project success.

What's Not Said About Developers

by Mayko Silva



Adding New Developers

Scrum Guide Flexibility

Scrum Guide doesn't specify timing for adding Developers

Anytime Addition

In practice, new Developers can be added any time during Sprint

Alignment with Scrum Philosophy

Aligns with Scrum's pragmatic and empirical approach

Impact on Team Velocity

Adding new developers to a team can have a significant impact on the overall velocity of the group. Contrary to what some might expect, bringing in additional manpower doesn't always lead to an immediate increase in productivity. In fact, it's quite common for team velocity to temporarily slow down when new developers join the team.

This temporary slowdown is primarily attributed to two main factors:

- The onboarding process
- The time it takes for new developers to become fully productive

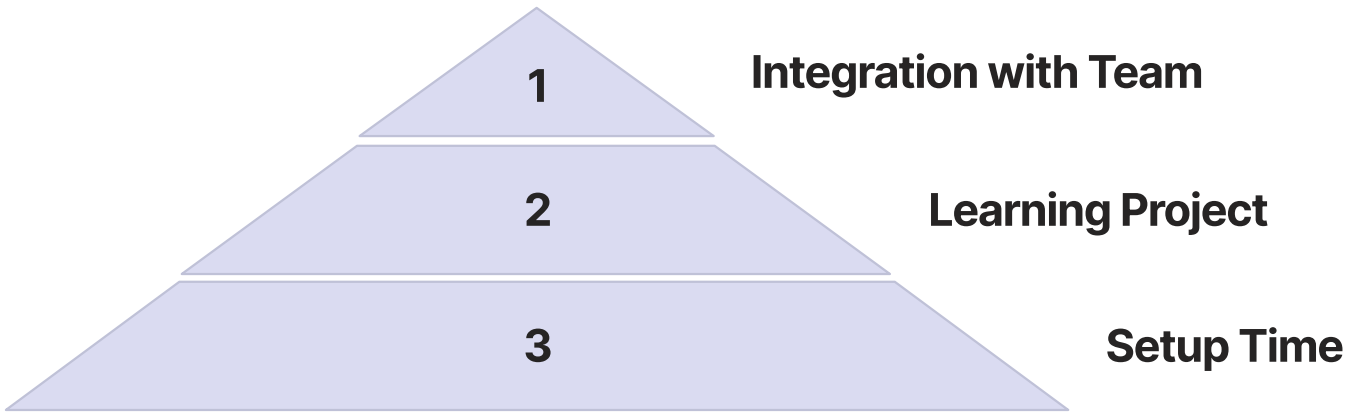
During the onboarding process, existing team members often need to dedicate time and resources to help integrate the new developers. This can include activities such as explaining project structures, coding standards, and team workflows. While essential for long-term success, these activities can initially reduce the team's overall output.

Furthermore, it takes time for new developers to reach their full productive potential within a team. They need to familiarize themselves with the codebase, understand the project's intricacies, and adapt to the team's working style. This learning curve can temporarily impact the team's velocity until the new members are fully up to speed.

Sustainable Pace

- Developers should **always** work at sustainable pace
- Unsustainable pace may indicate need for new Developers

Onboarding Process



New Developers need time to set up, learn project, and integrate with team. This process is crucial for their success and productivity within the organization.

It's important to note that existing team members often dedicate time to help onboard new Developers. This collaborative effort ensures a smooth transition and faster integration of new talent into the team.

Long-Term Productivity

When adding new developers to a team, it's important to understand the dynamics of long-term productivity. Initially, there's often a noticeable dip in productivity as the team adjusts to its new members. This is a natural part of the process and shouldn't be cause for alarm.

However, it's crucial to recognize that this initial decrease in efficiency is temporary. Over time, as new developers become more integrated and familiar with the project, the team's overall capacity should increase. This gradual improvement in productivity is the expected and desired outcome of expanding the development team.

The Path to Increased Capacity

While the short-term impact might seem counterintuitive, the long-term benefits of adding new developers can be substantial. As these new team members gain experience and become fully operational, they contribute to the team's collective knowledge and skills. This ultimately leads to enhanced problem-solving capabilities and increased output, boosting the team's overall productivity beyond its original levels.

Practical Example: E-commerce Platform

In this real-world scenario, we examine the mid-Sprint addition of a mobile optimization expert to an e-commerce platform development team. Initially, there's a noticeable slowdown in velocity as the team dedicates time for setup and knowledge sharing. To facilitate integration, the team employs pair programming techniques with the new developer.

Despite the initial setback, the new developer contributes valuable insights by the end of the Sprint. As the team adapts and synergies develop, subsequent Sprints witness a significant increase in velocity, demonstrating the long-term benefits of strategic team expansion.

Understanding Check

When is the best time to add new Developers to a Scrum Team?

A) Only at the start of a new Sprint
B) Only during Sprint Planning
C) Any time they are needed
D) Only after the Sprint Review

Answer and Explanation

Correct answer: C

- New Developers can be added any time they are needed
- Allows teams to respond to changing needs and maintain sustainable pace

Remember

Flexible Addition

New Developers can be added any time during Sprint

Temporary Slowdown

May temporarily slow down team velocity

Sustainable Pace

Goal is to maintain sustainable work pace

Team Effort

Onboarding requires time and effort from existing team

Long-term Benefit

Long-term result should be increased team capacity

Key Takeaway

Effective Team Scaling

Understanding unwritten aspects of Developer management is crucial for scaling teams efficiently and effectively.

Long-term Success

Grasping these unwritten aspects is essential for ensuring the long-term success of development teams and projects.

Adaptability and Value Focus

Comprehending these aspects allows teams to adapt to changing needs while maintaining a strong focus on delivering value.

Next Steps

Our journey doesn't end here. The next crucial step in maximizing value delivery is:

- Exploring effective Developer-Product Owner collaboration
- Understanding how this collaboration impacts value delivery

By focusing on this relationship, we can further enhance our team's productivity and ensure we're delivering the most value to our customers.