# PM² - Agile

Guide 3.0.1

**European Commission**

Centre of Excellence in PM² (CoEPM²)

# The PM²-Agile Guide 3.0.1

PM²

Brussels | Luxembourg, 2021

More information on the European Union is available on the internet (http://ec.europa.eu).

Luxembourg: Publications Office of the European Union, 2021

ISBN: 978-92-76-39410-5

doi: 10.2799/162784

**Title:** PM²-Agile Guide 3.0.1

**Current Edition:** PM²-Agile Guide, v.3.0.1 July 2021

© European Union, 2021

**Produced by:**

European Commission, DIGIT.B4.001 Project Shaping & Sourcing

**Published by:**

European Commission, DIGIT, Centre of Excellence in PM² (CoEPM²)

Authors and Contributions to this Guide:

See Appendix A: Acknowledgements.

Email: EC-Agile@ec.europa.eu

# Table of Contents

# 1 Agile Guide – Introduction

This guide introduces PM²-Agile, the Agile extension of the PM² methodology.

"Agile" is a collective term used to refer to a set of practices (driven by a set of core values and principles), especially in software development, in which requirements and solutions evolve through collaboration between self-organising, cross-functional teams. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change. To enhance these aspects, additional models and processes like the Lean Startup model and Lean UX (user experience) process have become essential tools to help people, teams and organisations succeeding with Agile.

Every year, several global surveys[1] confirm that Agile approaches yield impressive results. This is mainly because this approach makes it possible to manage changing priorities and leads to an increase in productivity and project visibility. The aforementioned surveys are very clear: the adoption of Agile is consistently growing worldwide.

However, Agile implementors face challenges related to:

- being Agile whilst complying with Enterprise processes, structures and rules, including IT governance and budgeting rules, programme structures, architecture and interoperability requirements.
- the application of Agile methods consistently across the organisation. Agile may not be as effective if some teams embrace an Agile approach and others do not.

PM²-Agile is designed to address these challenges. Moreover, incorporating Agile into the overall PM² framework, community and culture, creates the foundations for better organisational and personal learning and improvement.

**This guide is for:**

- Agile practitioners looking for a framework that could help them integrate their approach into existing structures and business rules.
- Organisations using PM² who want to start applying Agile values and principles to their way of working.
- Organisations' managers who need to know more about Agile, either for evaluation for their own use or to improve interactions within teams currently working in that way.
- Anyone interested in adopting Agile practices or just learning about them.

**This guide provides:**

- A common vocabulary to facilitate communication between and within project team.
- The PM²-Agile model.
- The Agile values and principles.
- The basics of Agile practices.

Summary information about some agile tools and techniques.

- A description of a set of recommended artefacts for documenting and reporting progress on activities managed using Agile principles.
- References and suggestions for further reading on Agile-related subjects.

The PM²-Agile guide primarily addresses the needs of software development projects. Agile approaches are also frequently applied successfully outside the software engineering environment. Therefore, most of the practices and tools and techniques described in this guide can also be used to non-IT projects. In addition, the (adapted) Agile values & principles have significantly influenced the PM²- Agile Mindsets that aim to guide project teams navigate the complexities of managing projects in their organisations.

This guide focuses on managing software development projects, in particular, how the work is organised. With a few minor exceptions, it does not cover the technical aspects of software production and implementation. In some cases, references to existing software engineering and Lean UX practices are also provided.

[1] VersionOne – Annual State of Agile Report;

PM²-Agile does not replace any part of PM². This guide, together with the rest of the PM²-Agile offering, expands PM² by providing specific methods, tools and techniques that help teams adopting and enabling the use of Agile and Lean principles in the context of their PM² projects.

These methods, tools and techniques are tailored to ensure the smooth flow of information, coherent structures, responsibilities and harmonised artefacts. PM²-Agile explains how to apply PM² to projects where at least one team, part of the Project Core Team (PCT), is applying Agile practices.

## 1.1 The Centre of Excellence in PM² (CoEPM²)

The purpose of the Centre of Excellence in PM² (CoEPM²) is to provide the European Commission and European Union Institutions with high-quality project management infrastructure, support and consulting services. The CoEPM² supports the PM² Methodology internally, coordinates an inter-institutional Project Support Network (PSN), and promotes the wider adoption of PM² through the opening of PM². PM²-Agile is also supported by the CoEPM² for the European Commission and the European institutions.

## 1.2 Opening PM²

PM² was made available by the European Commission, bringing the methodology and the overall PM² offering and its benefits closer to its broader stakeholders and user community. This increased effectiveness in the management and communication of project work serving the objectives of the European Union and needs of Member States and EU citizens.

Opening PM² aims to help avoid repeating mistakes of the past, such as replicating efforts and sponsoring divergent Project Management approaches based on differences rather than investing in converging approaches based on similarities and common interest of the broader EU Public Administration and beyond.

***"One common PM Methodology open to all EU Institutions, Member States, Service Providers, and EU Citizens."***

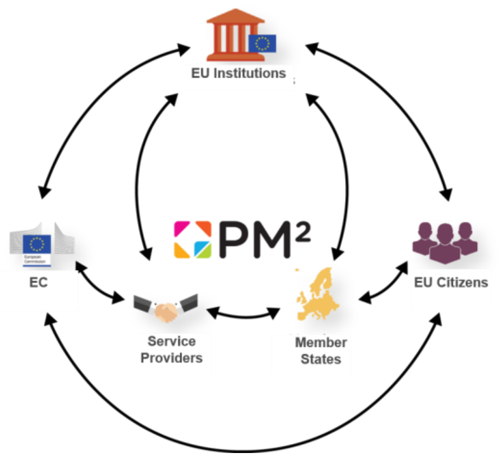


**Fig. 1.1** PM² Vision & Synergies

Opening PM² aims to contribute towards the increase in project management competency within broader user community and stakeholders and lead to increased project efficiency and success.

PM²:

- achieves rationalisation of project, programme and portfolio management approaches across the EU.
- establishes a common language & processes resulting in effective project communication.
- enables work transparency and visibility for cross-organisational and cross-border project collaborations.
- enables higher quality project, programme and portfolio management enabling cost/effort efficiency.
- enables better monitoring and controlling of EU funded projects and grants.
- materialises the European Commission decision of 12 December 2011 (2011/833/EU) on the *"reuse of Commission documents to promote accessibility and reuse."*

### 1.2.1 PM² Resources, Publications and Support

The CoEPM² provides a central online location for all PM² related information, publications, etc.

- PM² website https://europa.eu/pm2/

The CoEPM² also provides for the European Institutions advisory services, consulting and coaching on all PM² Methodologies.

- Contact EC-PM2@ec.europa.eu

### 1.2.2 Project Support Network

The PM² Project Support Network (PSN) is an internal network of Project Support Offices (PSOs), which are coordinated and supported by the Centre of Excellence in PM² (CoEPM²). The PM² Project Support Network (PSN) aims to become a decentralised Project Management support network, providing guidance and support to PM² users on both the PM² Methodology and the effective use of Project Management Tools & Techniques.

The Project Support Network (PSN) (internal):

- promotes the exchange and sharing of knowledge, experiences and best practices.
- makes it possible to collect feedback to continuously improve and build on the PM² Methodology.
- enables the Project Support Offices (PSOs) and Portfolio Support Offices (PfSOs) to support each other as a community.
- is coordinated and supported by the Centre of Excellence in PM² (CoEPM²).
- depends on the contributions of PM² champions (individuals and organisations).

The Project Support Network (PSN) (outside the European Institutions):

- promotes the exchange and sharing of knowledge, experiences and best practices with an emphasis on public services and institutions.
- aims to collect feedback to improve and build on the PM² Methodologies

Join the PM² Community and stay in touch for updates:

- https://joinup.ec.europa.eu/collection/pm2-project-management-methodology
- https://ec.europa.eu/eusurvey/runner/pm2-contact
- https://academy.europa.eu/courses/pm-c1-pm-essentials-project-management-methodology

This page has intentionally been left blank

# 2 About Agile

Agile is an approach described by a set of principles and values that focuses on delivering valuable solutions. It promotes adaptive planning, evolutionary development, early delivery, continuous improvement and encourages rapid and flexible response to change.

Agile appreciates the inherent uncertainty of the project environment and creates an organisation that is highly adaptive by using short feedback loops to quickly respond to changes in product requirements. It aims to constantly improve and streamline all project processes.

Key characteristics of Agile:

- Focus on delivering value early and frequently.
- Decisions are based on what is known.
- Close collaboration between all parties involved.
- Continuous stakeholder involvement at all levels.
- Plans are created with the involvement of team members.
- Incremental development with short cycles.
- Scope management by continuous (re)prioritisation of the Work Items.
- Embracing change, continuous learning and improvement.
- Just enough documentation and control.

Agile has its own roots but has been significantly influenced by the philosophy of Lean. The main principles of Lean are the transparency of the value stream and the reduction of waste. Lean identifies the following seven types of waste:

1. Inventory – unfinished goods (also called as "work in progress" - WIP).
2. Overproduction – producing more than demand requires.
3. Extra processing – additional steps in the process that are not really needed.
4. Transportation – shipping goods from one place to another.
5. Waiting – lag between process steps.
6. Motion – moving around within the process.
7. Defects – flaws in the deliverables that impact their features/functionality.

Agile approaches address these forms of waste in various ways. For example, work in progress (waste type 1) is reduced by having the right granularity of Work Items and by minimising dependencies. They go hand-in-hand, and together they ensure that to complete one piece of work, no other parallel work is needed..

Overproduction (waste type 2) can be due to extra features. This source of waste can be kept in check by having a well-prioritised Work Items List, interacting with the clients throughout the project lifecycle, and involving all team members in planning. Another form of overproduction happens when, at the end of any cycle, there is more product ready to be shipped than what can be effectively tested. This is minimised in Agile by the short feedback loop in the planning process, and having cross-functional teams and a transparent value stream. Overproduction is only one cause of process bottlenecks, and there are Agile techniques to help minimise these bottlenecks.

Non-value-added activities (waste type 3) and hand-offs (waste type 4) are minimised by introducing self-organising teams, cross-functional, collocated teams. This structure results in the emergence of effective, efficient and stable methods of working.

Waiting (waste type 5) and the frequent switching of tasks (waste type 6) are reduced by limiting the work in progress, delivering early and frequently, and ensuring clear priorities and an understanding of each work item.

Defects (waste type 7) are reduced by ensuring quality in the development process as opposed to investing in detection and correction activities. Apart from the overall Agile organisation work contributing to this, there are specific techniques like Test-Driven Development and pair programming[2] which help to lower the defect rate.

Reducing waste is only the first principle of Lean. All the others are also shared by Agile. However, although Agile is heavily influenced by Lean, it has its distinct features. For example, evolutionary development, cross-

[2] For more information on Pairing, please check the "Pairing" tool/technique

functional teams and the approach to internal and client collaboration are specific to Agile practices. These practices are based on the Agile values and principles described in section 2.1.

Agile promises higher chances of project and product success, higher employee and service provider engagement while resulting in software systems that provide a better user experience owing to the deeper involvement of stakeholders and way of accommodating their evolving needs.

Currently there are about a dozen Agile methods and many hybrids. Some of them have enjoyed serious take-up and media popularity. That has created a hype which has also proliferated many misconceptions. It is therefore important to clarify the Agile values and principles, and this clarification should guide the adoption of PM²-Agile over the currently popular practices and techniques.

## 2.1 Agile Values

In 2001, seventeen people brainstormed to find a better way to work, and from that emerged what they called the "Manifesto for Agile Software Development".

The result of their brainstorming session is published at http://www.agilemanifesto.org/. This content is usually known as the "Agile Manifesto" and defines four values supported by 12 principles.

The Agile Manifesto provides a foundation for the PM²-Agile framework. This foundation is adapted to reflect its values and the lessons learnt from years of Agile experience, as follows:

- The original manifesto focused on software development – PM²-Agile broadens the scope to focus on solution delivery.
- The original manifesto focused on customers – under PM²-Agile, all stakeholders are equally important.
- The original manifesto focused on development teams – PM²-Agile considers the overall organisational ecosystem and how to improve it.

Each of the four values statements in the Agile Manifesto is presented in the format "X over Y". The important thing to understand about the statements is that while you should value the concepts on the right-hand side, you should value the things on the left-hand side even more. A good way to think about the manifesto is that it defines preferences, not alternatives, encouraging a focus on certain areas but not eliminating others.

The four Agile Manifesto values are:

1. **Individuals and interactions** over **processes and tools**.
2. **Working solutions** over **comprehensive documentation**.
3. **Stakeholder collaboration** over **contract negotiation**.
4. **Responding to change** over **following a plan**.

#### First Agile value – individuals and interactions over processes and tools

Processes and tools are very useful but they are designed in a particular point in time and based on the knowledge available at that moment. However, because they are implemented and used at a later stage, they are often unable to accommodate the internal and external changes in projects. More importantly, work such as software development is based on people's knowledge and creativity.

Where processes and tools reach their limits, people's intelligence and ability to collaborate effectively can provide the necessary responsiveness and problem-solving capacity by generating new ideas and insights from the several different perspectives. As Agile recognises the inherent uncertainty of projects, it values individuals and interactions over and above processes and tools. This is one of the key enablers of the Lean UX process, an important pillar of PM²-Agile.

#### Second Agile value – working solutions over comprehensive documentation

The time spent documenting is time taken away from developing and learning by presenting prototypes of the project deliverables early in the project. Even if big part of an iteration deliverable is discarded, that effort is worthwhile for the feedback gained. And as this feedback often changes the specifications, there is all the more reason to prefer light documentation (or even better light modelling), because if most of what is specified changes, the specifying effort has been wasted. Moreover, you can often only discover the true requirements by combining the requesters' initial ideas with the realisation based on the developers' evolving understanding to produce something tangible, rather than just imagined.

Furthermore, stakeholders will have a much easier time understanding any working solution rather than complex technical diagrams describing its internal workings or describing an abstraction of its usage. Documentation has its place; deliverable documentation such as user manuals and operations manuals are in fact part of the overall solution.

The primary goal of any Agile Project Core Team (A-PCT) is to create solutions, not documents. That is why it is important to focus on the core deliverables.

There are three important points to make on this value in an organisational context.

- The original Agile Manifesto uses the term "software", not "solutions", for this value statement. Within an Organisation, the word "solutions" is preferred. This is to:
    - stress that the software is only one part of an IT project;
    - enable PM²-Agile to be applied to non-IT projects.
- Agile should be applied with the understanding of the core deliverable(s). In cases when Agile is applied to non-IT projects, it could be that the core deliverable is a document. In that case, "documentation" should be understood as that type which supports the production of the core deliverables.
- There is a risk of "working solutions" being interpreted in a way that leads to the development of locally optimal solutions, or "local optima" (rather than globally optimal solutions). Therefore, projects that are regarded as successful sometimes contribute to the duplication of IT investments (building a capability which is available from an existing application) or are designed in such a rigid, closed and non-interoperable way that major additional investment is needed to integrate that application with others.

**Third Agile value – stakeholder collaboration** over **contract negotiation**

Contracts are often signed early, leading to burdensome change management procedures. As change is inevitable, it is better to handle it by collaborating effectively with stakeholders. Furthermore, contracts bring a false sense of security, while the realisation that the knowledge of what is required and what has to be done relies on collaboration to solve the issues and take advantage of opportunities not envisaged in the contract.

The stakeholder collaboration generates a system of consensus-based decisions that allows creating a common and shared understanding of the problem to be solved and the possible solutions.

(Note that the original Agile Manifesto used the term "customer" instead of "stakeholder" for this value statement).

**Fourth Agile Value – responding to change** over **following a plan**

It is very important to have a plan, but it should only be detailed for the current iteration. This will make it possible to make adaptations based on feedback received during the current iteration and plan for the next one. Agile practices do allow for longer-term planning, but these plans should be high-level and created on the understanding that they may need to be updated frequently. This fourth value, at its core, is the key enabler of the Build-Measure-Learn feedback loop from the Lean Startup model, another important pillar of PM²-Agile.

## 2.2 Agile Teams

The most important productivity factor in any team are the people and the way they interact. Agile teams are self-organising, cross-functional, fluid, and highly collaborative.

Self-organisation[3] means that everyone in the team works together to determine the best way to perform the work. The team is cross-functional, containing members with the needed combined expertise to build and deliver a solution. This includes people with various technical skills. It also includes stakeholders with the required domain knowledge. Fluidity refers to the idea that the team composition may vary over time.

[3] For more information on Self-Organising teams, please check the "Self-Organising Teams" tool/technique

Though the PM²-Agile roles have some specific responsibilities, all project team members for a project following PM²-Agile have some shared responsibilities.

- Deliver the outputs that best meet the stakeholders' expectations and respect the organisation's quality standards.
- Respect the commitments.
- Respect the project budget and strive to be efficient while delivering the outputs.
- Promote a collaborative and cooperative working environment.
- Share information, knowledge, and expertise with colleagues.
- Help other team members develop their skills and competencies.
- Be proactive and have a "grow" mindset, internalising a continuous learning and development approach, be it on soft-skills or hard-skills.
- Contribute to the continuous improvement of PM²-Agile.

Agile teams work in a highly collaborative[4] manner, adopting the most effective communication techniques for their situation and striving to work together as closely as possible.

The goal is to ensure that:

- everyone has a sense of belonging on the team, of being in it together. There should be no "outsiders", no "them" but only "us";
- the team includes everyone required to build the system. A self-contained team should have the skills and knowledge to get the job done. This is not always possible, and sometimes there is a need to bring in outside experts for brief periods of time for specific tasks;
- everyone in the team contributes any way they can. There is a move away from specialists who focus on a specific category of work, such as analysis or database administration, towards teams with multi-disciplinary members;
- the team is self-organising. The people best-suited to plan and organise the work are the ones who do the work. This results in better estimates, more realistic schedules, and increased commitment to the plan by the team;
- the team maintains a sustainable pace. No heroic contributions, no burnouts;
- everyone works together closely.

## 2.3 Continuous Improvement & Validated Learning

A core philosophy of Agile is that of continuous improvement and learning. This is done through frequent reviews and retrospectives, which are described as rituals where teams learn from the experience and plan changes for the next cycle. Reviews are a great tool to collect feedback from the stakeholders and learn about the solution. Retrospectives are a great tool to collect feedback and learn about the process and the Agile Team itself. Both are key in delivering a successful project and solution as they will guide the overall improvement process. However, this could not be achieved without effective and meaningful learning. PM²-Agile considers this a cornerstone and embraces the concept of Validated Learning, a part of the Lean Startup model, which also plays an important role in PM²-Agile.

While the Lessons Learned captured at the end of a project remain valuable, the spirit and habit of frequently reviewing and retrospecting must be infused across the project continuum, since this is where continuous improvement and validated learning take place.

**Build-Measure-Learn**

The teams follow a feedback loop to continuously learn from their hypotheses and improve by taking advantage of the triggers: the ideas, the product and the data.

The Build–Measure–Learn loop emphasizes speed as a critical ingredient to continuous and validated learning for proper development afterwards. A team's effectiveness is determined by its ability to ideate, quickly build (rapid prototype) a minimum viable product of that idea, measure its effectiveness in the market, and learn from that experiment by analysing the generated data. In other words, it's a learning cycle of turning ideas into products, measuring users' reactions and behaviours against built products, and then deciding whether to persevere or pivot the idea; this process repeats as many times as necessary.

[4] For more information on Teamwork, please check the "Assessing Teamwork" tool/technique

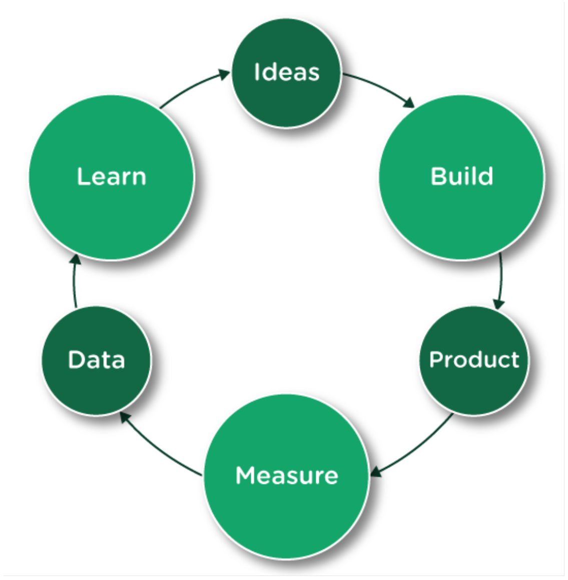


**Fig. 2.1** Minimize total time through the loop[5]

**Minimum viable product (MVP)**

Building Start-ups involves many risks and timing. It is safer for start-ups to experiment on their idea by preparing early prototypes and test them with their potential clients as early and continuously as possible. They came up with the idea of an MVP since there was not much time to build something cheap and fast that would allow them to gain learning and lead the market.

Building an MVP doesn't necessarily mean that a fully functional piece of coded software needs to be developed. The most important aspect is that the usage of the MVP can be measured and the data collected is used to learn and generate new ideas. This is what will keep the cycle going and consequently, make the solution grow towards the right solution. The Lean UX process (another cornerstone of PM²-Agile) goes hand in hand with the Lean start-up model by helping to generate the MVP with rapid prototyping and gain fast learning.

The Lean start-up model and Lean UX process have been implemented also by the public sector worldwide to reduce the uncertainty of costs and risks at later stages of product development, where many stakeholders and governmental budgets are involved.

The Minimum Viable Product (MVP - figure 2.2), is the result of continuous validated learning that gives the green light for Incremental Development or even an indication of a non-continuity of an idea.

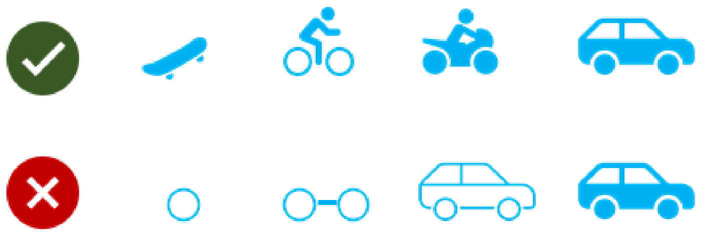


**Fig. 2.2** What an MVP is and what it is not

**The Purpose of an MVP**

- Be able to test a product hypothesis with minimal resources.
- Accelerate learning.
- Reduce wasted engineering hours.
- Get the product to early customers as soon as possible to learn from its use.
- Base for other products.

[5] Adapted from: *The Lean Startup*: Eric Ries – Currency International Edition 2017

Figure 2.3 shows all the key elements that must be considered to build the right product with fewer risks and be ready for the market. It is highly recommended to incorporate all those elements instead of shortcutting and focus on just one or two.

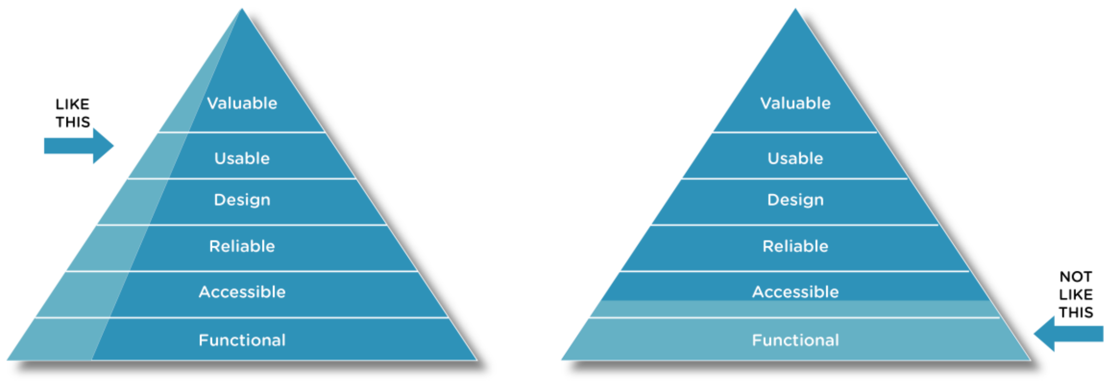


**Fig. 2.3** The right MVP: all key elements are met

An MVP must include and focus on the following key elements:

- Functionality: the set of features must deliver/demonstrate clear value to the user.
- Accessibility:  the MVP must comply with accessibility standards.
- Reliability: the adequate evidence that confirms that the initial problem or goal is solved in a manner that it is coherent and relevant to move forward.
- Design: the design of the MVP must be up to the highest industry standards.
- Usability: the MVP must be easy to use and intuitive.
- Value: the MVP brings organisational value and serves user needs.

**Success criteria.**

- Comply with all key elements.
- Define from the beginning key performance indicators (KPIs) and metrics.
- Conduct frequent user research with users and validate with stakeholders. Try to keep up the research up to five users to avoid wasted time with over-analysis.

## 2.4 Agile Myths

Some people (both IT and business-oriented) are very enthusiastic about applying Agile approaches to their work. Others are sceptical about Agile for a number of reasons, influenced perhaps by certain myths about Agile. Some of these myths are listed below:

**Myth — Agile is a silver bullet**

- You can fail as in any other project. But with Agile you might fail sooner and in doing so, you save resources.

**Myth — Agile teams do not do documentation**

- They do not produce *unnecessary* documentation.

**Myth — Agile is anti-planning**

- In Agile development there is extensive planning on multiple levels, and with high visibility.
- There is a Release Plan, Iteration plans, and daily work is planned during Stand-up Meetings.
- There are different tools for planning (e.g. product and iteration backlogs).

**Myth — Agile is undisciplined**

- In fact, Agile development is extremely disciplined. Practices like Continuous Integration, Automated Testing, Iteration reviews and retrospectives, Iteration and release planning, etc, are some examples of how disciplined Agile development must be.
- In most Agile practices all iterations and meetings are timeboxed.
- There is commitment to "ship product" regularly.
- Improving the practice is part of the practice.

**Myth — Agile is Scrum.**

- Scrum is not the only Agile approach, there are at least ten more.
- Scrum focuses mostly on the "construction" aspect of projects.
- Scrum does not address activities for managing the complete project lifecycle.
- Scrum has become one of the most popular Agile frameworks (true).

**Myth - Agile is anti-architecture**

- Solution architecture is minimised to what is known.
- Solution architecture is valuable and in one or another way it is used by Agile teams.

This page has intentionally been left blank

# 3 PM²-Agile – Overview

PM²-Agile is an offering that aims to enable and support the use of Agile practices in any type of project or work activity. Although such practices may already exist, ensuring the success of the project may be difficult due to the scale and interdependencies in the organisation's ecosystem.

Unlike practising Agile in small organisations, large organisations may need to enable collaboration between Agile and other project approaches while having to comply with various IT governance, enterprise architecture and interoperability requirements. PM²-Agile is designed to provide this support both to current practising Agile practitioners as well as to those who are willing to try Agile approaches to their projects.

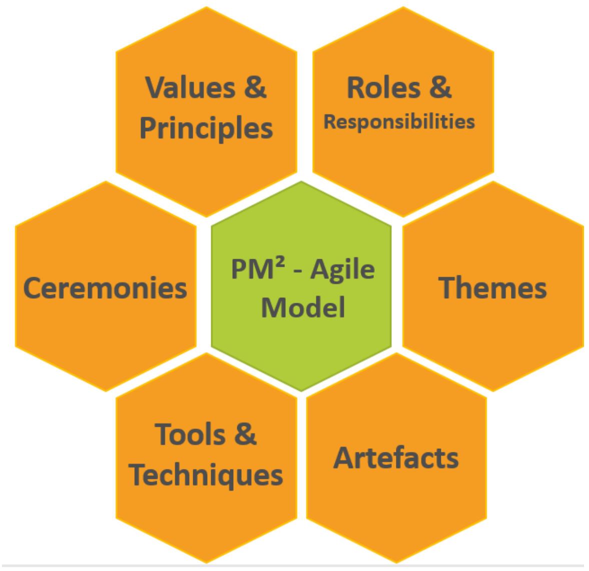


**Fig. 3.1** The elements of PM²-Agile

## 3.1 PM²-Agile Principles

To help people gain a better understanding of what Agile software development is all about, the seventeen authors of the Agile Manifesto described what they mean by "Agile" in 12 principles. PM²-Agile has adopted – with some modifications – these principles.

The 12 principles of PM²-Agile:

- The highest priority is to satisfy the client through early and continuous delivery of valuable solutions.
- Changes in requirements are welcome.
- Deliver value frequently through working solutions.
- Business people and the Project Core Team (PCT) must work together throughout the project.
- Create teams with motivated individuals. Give them the environment and support they need to self-organise, and trust them to get the job done.
- The most efficient and effective method of communication is face-to-face conversation.
- The primary measure of progress is the value and use of what has been delivered.
- Continuous attention to quality.
- Simplicity – the art of maximising the amount of work not done – is essential.
- At regular intervals, the team reflects on how to improve, then tunes and adjusts its behaviour accordingly.
- Agile processes promote sustainable development. The Project Team should be able to maintain a constant pace indefinitely.
- Agile practices should be enterprise-aware by considering IT governance, enterprise architecture, and interoperability requirements. Agile teams should be able to collaborate effectively with teams and stakeholders following alternative approaches.

You can read more about the PM²-Agile Principles in *Appendix B: Additional Resources*.

## 3.2 Agile & PM²

PM² is the official Project Management Methodology of the European Commission (EC). Its purpose is to enable Project Managers (PMs) to deliver solutions and benefits to their organisations by effectively managing the entire lifecycle of their project. PM² has been created with the needs of European Union Institutions and projects in mind, but it's transferrable to projects in any organisation. It is a well-balanced approach, enabling delivery teams to work autonomously while ensuring conformity with the enterprise structures and operations. PM² has built-in flexibility and does not prescribe specific processes at the technical level. This approach enables the rapid adoption of Agile practices. Additionally, Agile teams can fully account for higher-level governance constraints by aligning with PM².

IT projects must consider their organisations' policies, decisions and communications. Other factors to consider include:

- a fixed budgeting cycle;
- a traditional command and control management culture;
- a traditional waterfall project lifecycle;
- a functional/weak matrix project organisation;
- clear project-approval milestone;
- upfront planning and architecture decisions.

These organisational factors, when accompanied by best-practices and activities (defined locally at an organisational/departmental level), can build a robust framework in which IT projects can be delivered. The clear set of roles and responsibilities also complements these factors thereby allowing team members to assume specific and well-defined responsibilities during the lifecycle of a project.

The PM² Project Management Methodology includes and discusses solutions for the above-mentioned constraints.

PM² is a light and easy-to-implement methodology which project teams can tailor to their specific needs.

**The PM² Methodology provides:**

- A project governance structure
- Process guidelines
- Artefact templates
- Guidelines for using the artefacts
- A set of effective mindsets

PM² acknowledges the complex and uncertain nature of many types of projects and the positive contribution of the Agile attitude to their effective management. However, certain management requirements cannot be fully satisfied by using a purely Agile project management approach.

PM² is a comprehensive project methodology supported by a project governance structure with a serial phase lifecycle. It covers the full breadth of management and project delivery while endorsing an enterprise-aware Agile approach. The seamless integration of PM² and Agile helps teams to be policy-, process- and audit-compliant. While accommodating audit requirements, working with contractors, collaborating on other projects and with other departments, and coordinating with programme and portfolio groups, PM²-Agile empowers project teams to achieve the desired 'agility'.

**PM²-Agile provides:**

- Roles & Responsibilities (as an extension to the PM² Governance)
- Integration with the overall PM² project lifecycle
- Core Themes
- A key set of Ceremonies
- A set of suggested Agile artefacts (as an extension to the PM² Artefacts)
- An Agile glossary
- Agile Tools & Techniques

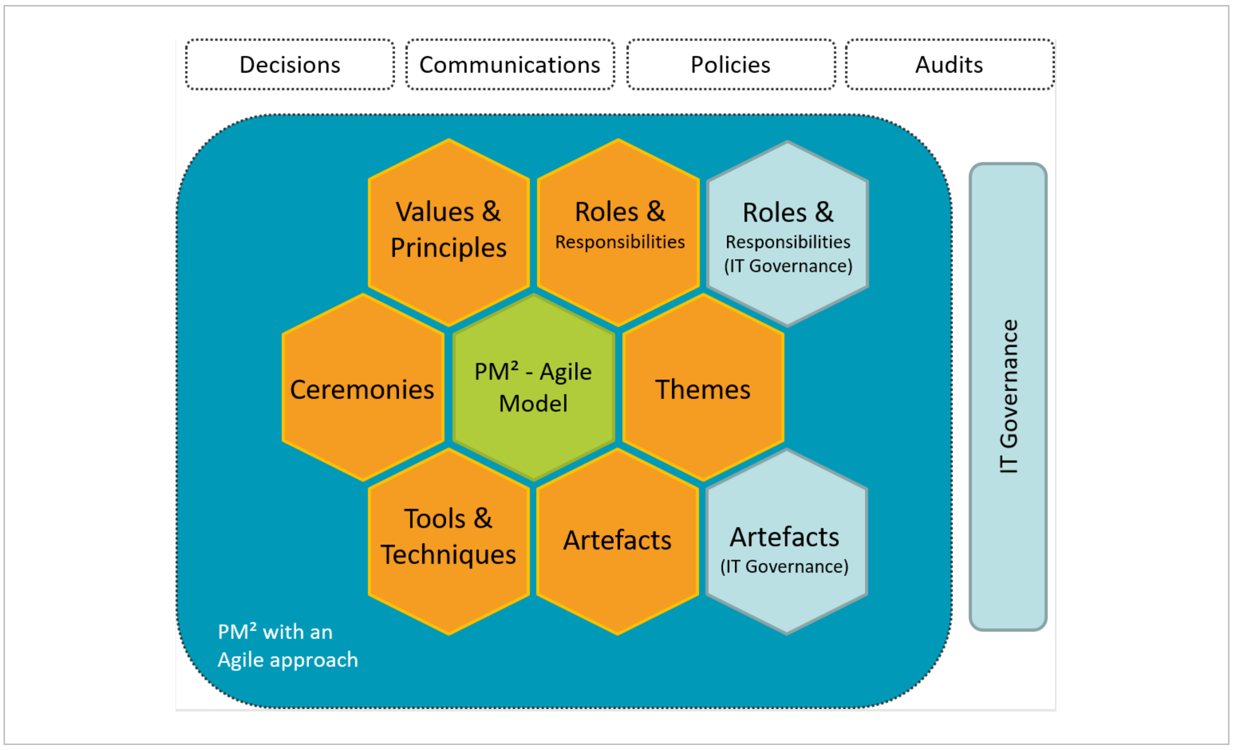


**Fig. 3.2** PM²-Agile elements within the context of PM²

When Agile is chosen for a PM² project, the incremental and iterative development approaches enable a new delivery strategy, which is materialised with the help of the PM²-Agile specific elements described in figure 3.2.

Because it's a PM² project, all these specific elements are framed by the PM² methodology, ensuring a solid connection throughout the entire project lifecycle via the governance model (Roles & Responsibilities) and the artefacts.

## 3.3 The PM²-Agile Model

PM²-Agile is not a new Agile philosophy or approach. PM²-Agile simply tailors proven Agile and Lean practices and connects them to PM². PM²-Agile allows Agile teams to remain aligned with PM² Project Management Methodology by enabling them to apply Agile values and principles and maintain enterprise awareness and compliance at the overall project level.

Figure 3.3 shows how PM²-Agile is connected to Project Management, Software Engineering and the other Enterprise Practices (e.g. Portfolio Management, Programme Management, IT operations etc.).

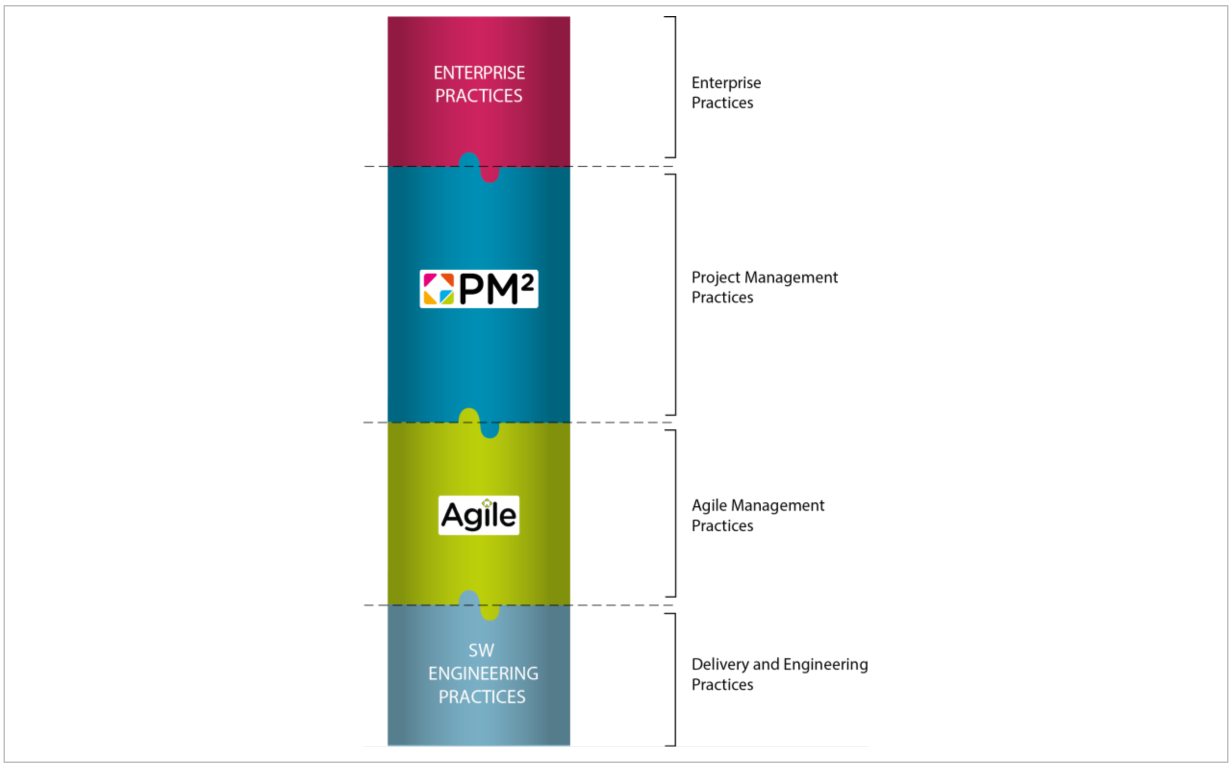

ENTERPRISE PRACTICES
PM²
Agile
SW ENGINEERING PRACTICES
Enterprise Practices
Project Management Practices
Agile Management Practices
Delivery and Engineering Practices

**Fig. 3.3** The positioning of PM²-Agile within the Organisation environment

Among the advantages over other popular standard Project Management methodologies, PM² consciously enables ‘agility’ (i.e. the PM² processes and mindsets are aligned with the Agile philosophy) and interlinks with the rest of an organisation’s management structures and processes. Though PM²-Agile is not a software engineering library, it does include some optional techniques such as Test-Driven Development, Pairing, etc. However, since it is used primarily for software development, PM²-Agile is well-integrated with software engineering practices.

Agile advocates self-organising teams. PM² provides the appropriate governance framework, which reflects the governance needs of the organisation as a whole, but also enables the Project Core Teams (PCT) to harness the benefits of self-organisation. For this purpose, PM²-Agile organises Project Core Team (PCT) members who work in an Agile way.

Figure 3.4 below shows how PM²-Agile teams fit into the overall project organisation

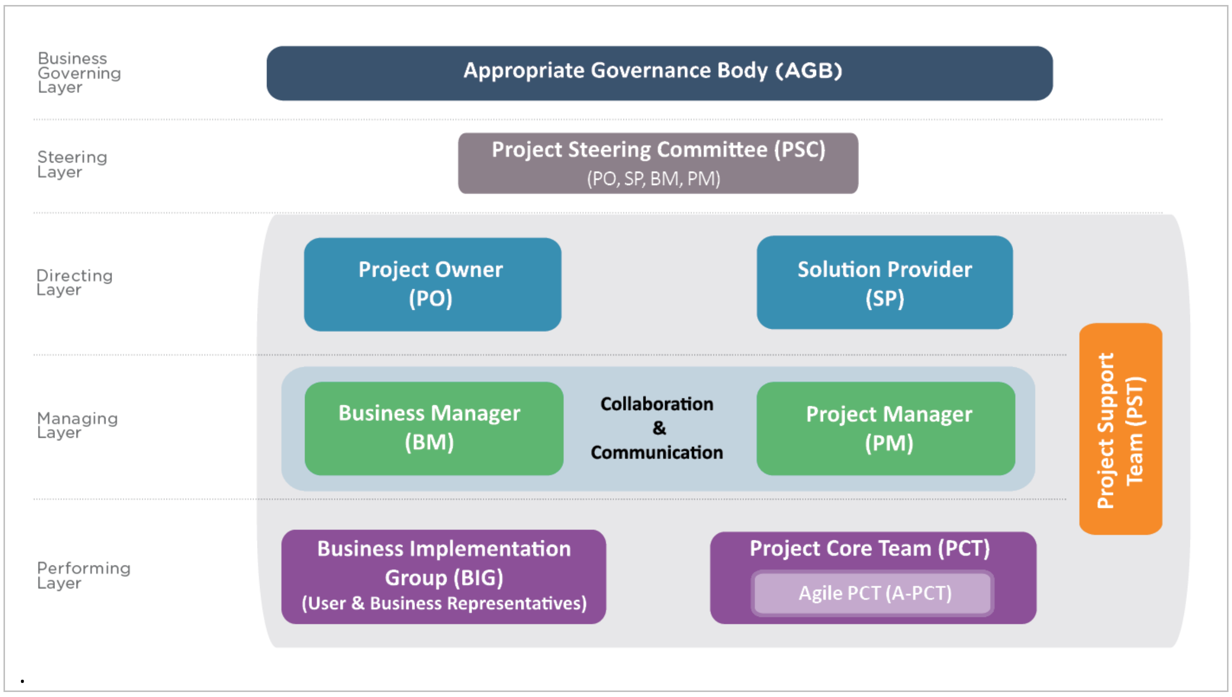


**Fig. 3.4** Agile Teams in the context of the PM² project organisation

A set of roles and responsibilities is set out for the Agile team (see section 6, “Roles and Responsibilities”).

### 3.3.1 Lifecycle

Every project has a beginning and an end. The lifecycle of a project has an identifiable start and end, which can be associated with a time scale. Each project phase represents a period of time during the life of the project where similar types of activities are executed (e.g. planning type activities ‘peak’ in terms of effort during the planning phase, etc.). Projects pass through four distinct phases, as described by PM².

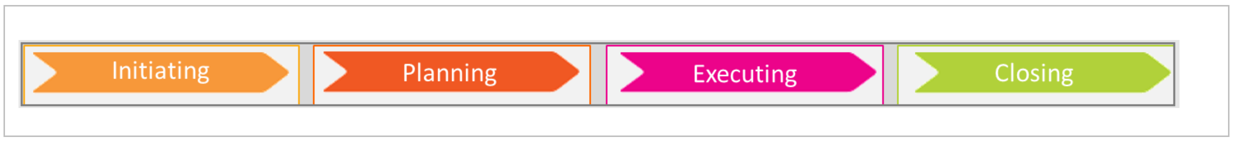


**Fig. 3.5** PM² project phases

The project lifecycle shown in Fig. 3.5 includes all events and activities from the point of inception of the idea/need through to the final completion of the project. Note that the interfaces between phases are seldomly separated clearly – activities related to a specific phase (e.g. planning activities) continue to be executed during the subsequent phase(s) (e.g. during the executing phase).

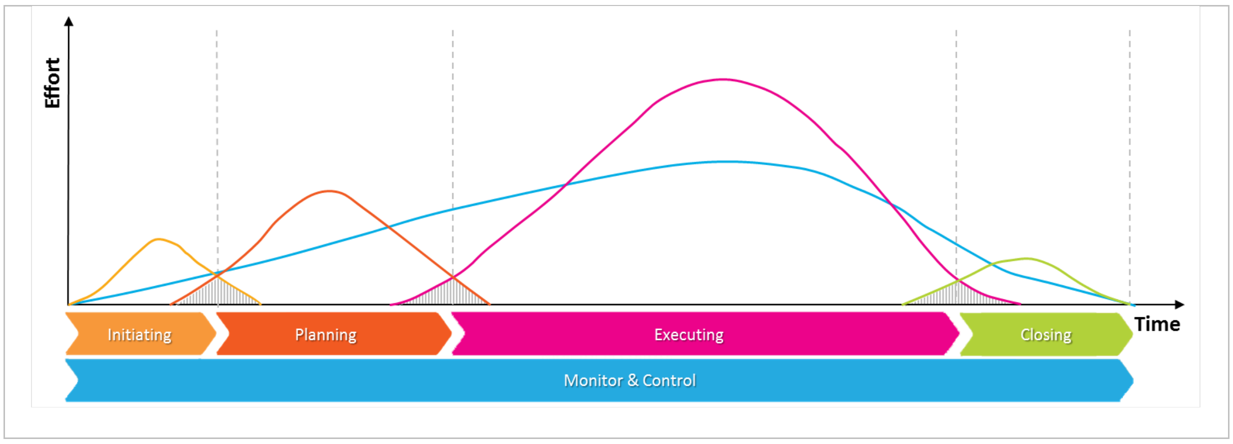


**Fig. 3.6** The PM² project lifecycle – overlapping of phase-related activities

The characteristics of some of these phases (e.g. planning) may not be as tangible at the level of the Agile team as they are at the overall project level. However, project phases still exist and a common understanding between both the project team members and other project stakeholders should be fostered. The overall sequential character follows the concept that the project has a defined beginning and ending date. Therefore, project activities must start with initiating and planning, then lead to implementation, and finally end with acceptance, transitioning and closing. However, team-level tasks do not have a sequential character within iterations: planning, analysing, coding, testing and reviewing are done in every iteration. Some Agile practices may also include a special time slot for planning and reviewing.

The PM²-Agile has iterative cycles at three levels – daily cycles, iterations, and releases. Regardless of their duration, these cycles follow what is known as the PM²-Agile ‘CIR’ rhythm (Coordinate, Implement, Review).

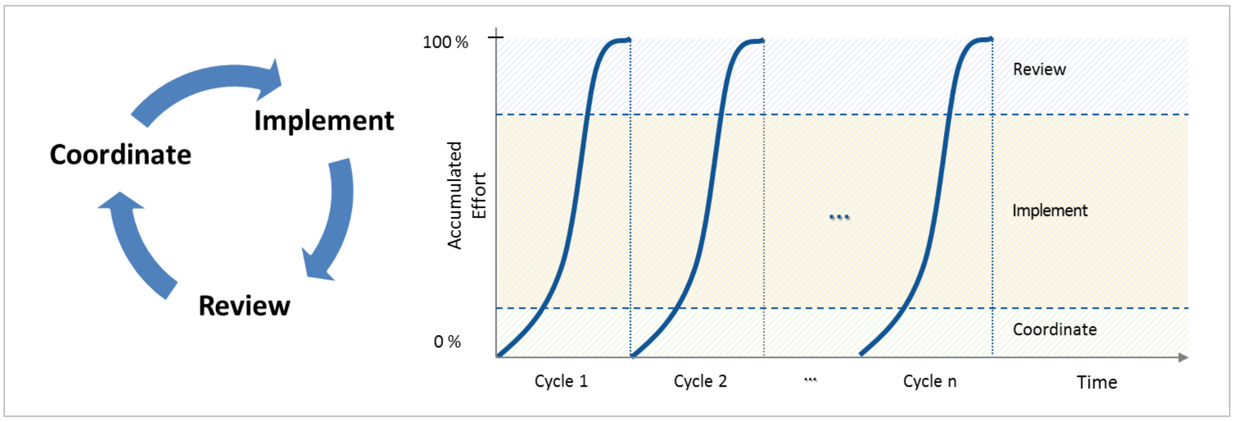


**Fig. 3.7** The PM²-Agile CIR rhythm

The combination of the high-level phases and lower level iteration cycles results in an effort/time curve with greater overlap between activities over time, spanning multiple phases of the project.

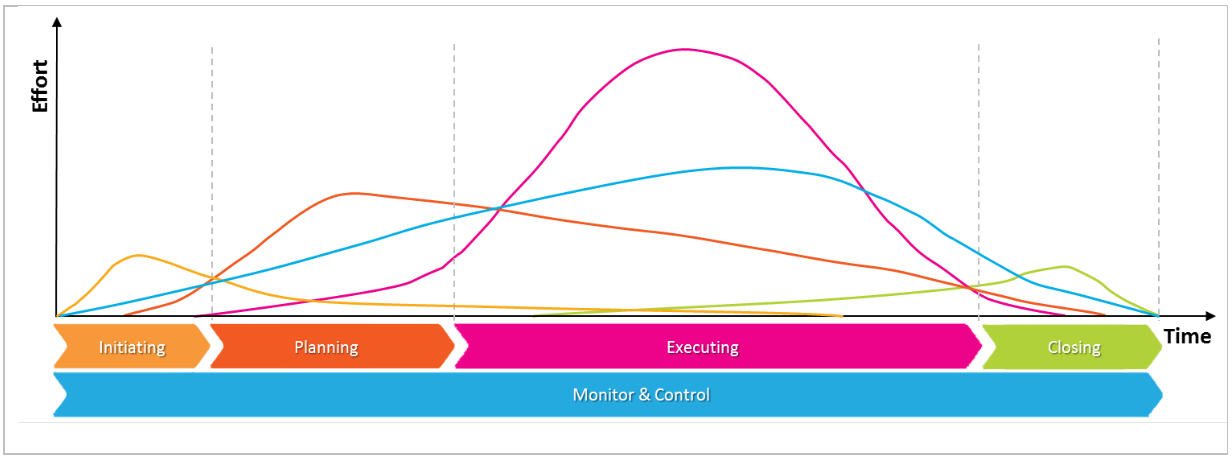


**Fig. 3.8** The PM² project lifecycle with activity iterations – extensive overlapping of phase-related activities

At the beginning of the project, a key concern is to understand what to build by determining an overall vision and identifying the stakeholders and their success criteria. Because there is a lot of uncertainty at this stage, PM²-Agile acknowledges the need to identify and declare all the assumptions that will guide the identification of the possible system capabilities. Declaring assumptions and not treating them as facts, gives the stakeholders and the team the opportunity to ask questions and discuss about the problems and the best way to solve them. The goal is to ascertain at least one solution and then assess whether it is technically feasible based on those assumptions. This is done by identifying the several hypothesis that must be tested in order to validate the declared assumptions and mitigate the uncertainty risks as soon as possible.

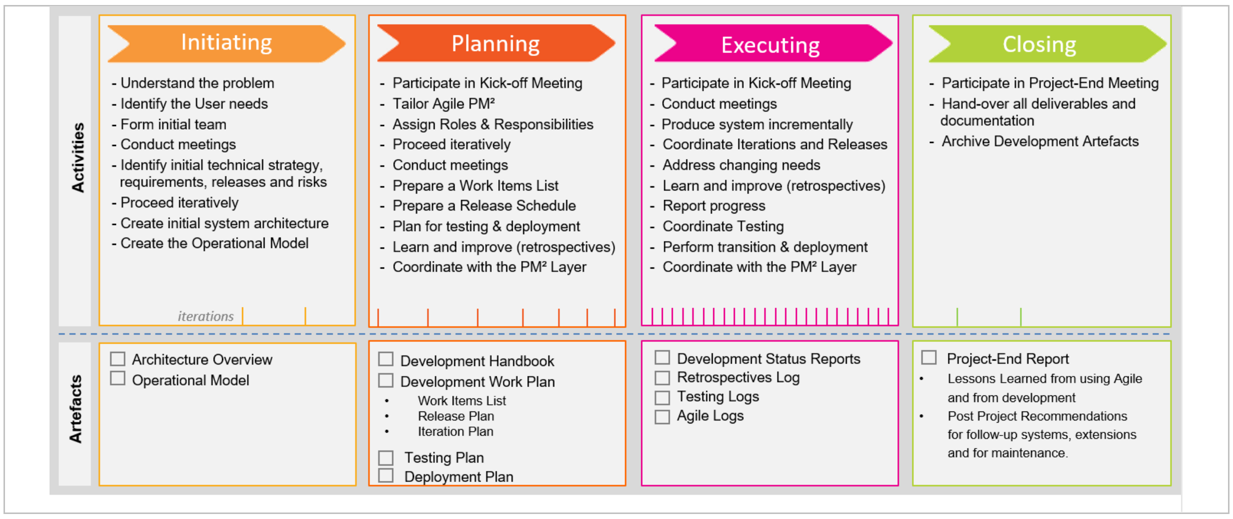
It is also critical to understand the high-level estimates of costs, schedules, and risks associated with the development aspect of the project. Including these estimates in the project-level estimates and planning artefacts ensures that they are considered.

As the project progresses, the focus of the domain-specific activities shifts towards incrementally building the solution. As the team progresses towards testing the hypotheses that validate the declared assumptions, a better understanding is achieved of what has to be done and how. PM²-Agile supports these actions by incorporating a benefit hypothesis for each feature that targets the validation of an assumption.

As the development progresses throughout the different iterations, requirements are continuously revisited and clarified. Both technical and non-technical risks are also regularly addressed. This iterative and incremental nature of PM²-Agile creates a sustainable and paced delivery of value to the organisation. This value is measured rigorously using a measuring and learning framework. This framework captures proper and relevant data from an MVP (or subsequent versions) and uses it as a (validated) learning mechanism to enable decision-making.

As the project nears completion, the focus shifts to ensuring that the solution is well-integrated into the overall project deliverables. Furthermore, it must contribute to the accrual of the benefits desired by the organisation.  There is also an emphasis to ensure that the organisation can assume ownership and is accountable for the evolution and maintenance of the information system when the project closes.

A summary representation of the key Agile activities and artefacts for each project phase is shown in figure 3.9. A detailed presentation of the Agile artefacts is provided in section 7, 'Artefacts'. Note that artefacts are delivered incrementally and iteratively following the Agile principles.



| | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Activities | - Understand the problem<br>- Identify the User needs<br>- Form initial team<br>- Conduct meetings<br>- Identify initial technical strategy, requirements, releases and risks<br>- Proceed iteratively<br>- Create initial system architecture<br>- Create the Operational Model<br>iterations | - Participate in Kick-off Meeting<br>- Tailor Agile PM²<br>- Assign Roles & Responsibilities<br>- Proceed iteratively<br>- Conduct meetings<br>- Prepare a Work Items List<br>- Prepare a Release Schedule<br>- Plan for testing & deployment<br>- Learn and improve (retrospectives)<br>- Coordinate with the PM² Layer | - Participate in Kick-off Meeting<br>- Conduct meetings<br>- Produce system incrementally<br>- Coordinate Iterations and Releases<br>- Address changing needs<br>- Learn and improve (retrospectives)<br>- Report progress<br>- Coordinate Testing<br>- Perform transition & deployment<br>- Coordinate with the PM² Layer | - Participate in Project-End Meeting<br>- Hand-over all deliverables and documentation<br>- Archive Development Artefacts |
| Artefacts | ☐ Architecture Overview<br>☐ Operational Model | ☐ Development Handbook<br>☐ Development Work Plan<br>• Work Items List<br>• Release Plan<br>• Iteration Plan<br>☐ Testing Plan<br>☐ Deployment Plan | ☐ Development Status Reports<br>☐ Retrospectives Log<br>☐ Testing Logs<br>☐ Agile Logs | ☐ Project-End Report<br>• Lessons Learned from using Agile and from development<br>• Post Project Recommendations for follow-up systems, extensions and for maintenance. |

**Fig. 3.9** An overview of the PM²-Agile activities and artefacts in each phase

From an Agile perspective, projects may have several releases as a result of the incremental progress achieved in one or more iterations. Additionally, the output of each iteration is the result of the incremental progress achieved every day. This multilevel approach, as shown in figure 3.10, spans the several phases of the project as foreseen in PM² and at all three levels, value is delivered and feedback is collected.

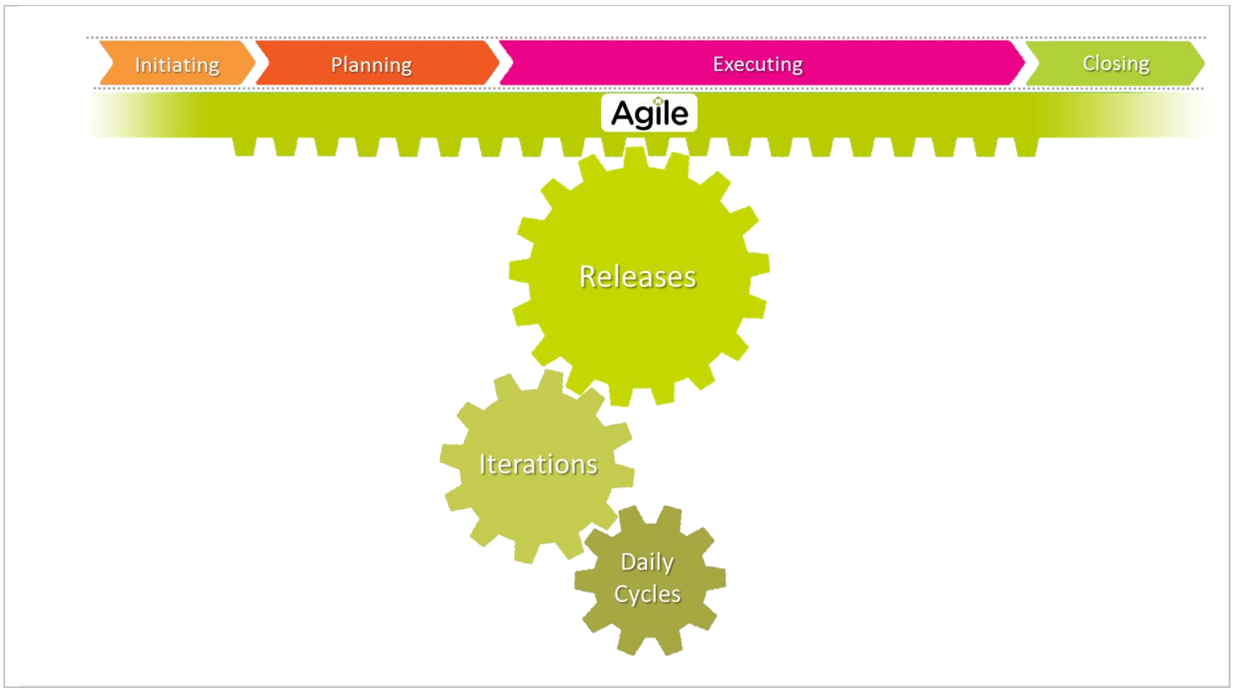


**Fig. 3.10** From project phases to daily cycles

### 3.3.2 Iterations

An Agile iteration is a period of time within a project in which the Agile Project Core Team (A-PCT) produces a stable, potentially shippable part of the solution, together with any other necessary supporting documentation or deliverables. Iterations are timeboxed, meaning the iteration duration is fixed, and the scope of the iteration's content is actively managed to meet that schedule.

Each iteration produces a product increment which brings the system one step closer to the final product. Within each iteration, prototypes or specific parts of the system are developed or updated. It is a bit like 'growing' software. Instead of developing the system one part after another, the whole system is evolving through each cycle using the CIR rhythm.

Each timeboxed iteration is properly planned. The iteration goals are well-defined; the acceptance criteria are established; and the tasks and responsibilities of participants are made clear. Additionally, progress transparency is assured through agreed-upon metrics and methods for measuring.

Each iteration must dedicate a portion of its time to lay down the foundation for the work to be accomplished in the following iteration. This is where Design Blocks (a part of the Release Planning ceremony) play an important role, allowing teams to challenge the requirements that were received from the Product owner (PrOw) and other Business stakeholders. Within Design Blocks, Agile Team Members (ATeMs) facilitate a set of workshops that focus on sketching and developing low fidelity prototypes for the future features until a common understanding is reached. High fidelity prototypes are then produced and delivered so they can be presented by the Product Owner (PrOw) to the business stakeholders for validation, like the Business Manager (BM) or the Buisiness Implementaion Group (BIG). When the Agile Project Core Team (A-PCT) is confident and agrees upon which elements to deliver first, development can then start.

Each iteration should implement the highest-priority Work Items and address the most critical risks. This ensures that every iteration adds maximum value while reducing uncertainty. Incremental and Iterative development are typically combined with Continuous Integration. As unit-tested components become available, they are integrated to generate a build which is then submitted for integration testing.

Therefore, the capability of the integrated software grows as the iteration proceeds towards the iteration goals. Regular builds (such as daily or more frequent builds), make it possible to break down the integration and test issues and spread them across the development cycle. Often, the downfall of large projects is caused because all the problems are discovered at the same time during the single massive integration that occurs very late in the project cycle. The later in the project cycle, the higher is the risk that a single problem can halt the progress of the whole team using a non-Agile approach.

The best and most effective way for an Agile Project Core Team (A-PCT) to demonstrate tangible progress is by exposing the output generated in each cycle. This enables teams to get feedback at an early stage so that they can continuously improve their understanding of what needs to be done and how to do it.

In PM²-Agile, three ways of exposing work are considered. While the goals of these methods are similar (the ability to demonstrate progress, deliver value and collect feedback to reduce uncertainty), the cadence and the formalism required are slightly different.

- **Exposure of Work Items** – This occurs when the team completes one or more Work Items from the Work Items List.
- **Exposure at the end of an Iteration** – This occurs at the end of the Iteration and supports the Iteration review ceremony. It includes all the Work Items that are deemed as ready for review during that iteration.
- **Release** – This is an agreed Solution Increment, delivered to the client and users so that they can start using it as a real product. It includes all the Done Work Items from several prior Iterations.

The following sections provide additional information about each of these approaches.

**Exposing Work Items**

The key driver of an Agile Project Core Team (A-PCT) is the Work Items list, prioritized by the Product Owner (PrOw). When the team completes an item from the list, some tangible progress is ready to be delivered to the client and valuable feedback is waiting to be collected. This cadence should be preserved throughout the iteration until the team presents the accumulated work during the Iteration Review, as demonstrated in figure 3.11.

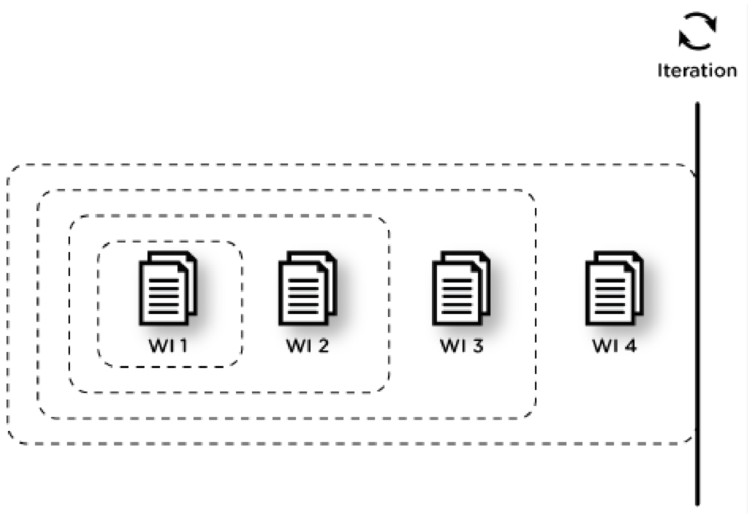


**Fig. 3.11** Exposing Work Items

Releasing a work item should be a fairly simple and straightforward process. A simple deployment in a standard test environment is often sufficient.

After exposing the Work item, the Agile Team Members (ATeMs) involved in its implementation will notify the Product Owner (PrOw) and request feedback. Because it's a recurrent process, it's important to keep it light and the formalism kept to a minimum, as the goal is to quickly collect feedback as a preparation for the Iteration Review. In the end, it's the "sum" of all Work Items exposed that will make the Iteration Review a great Iteration Review.

**Demonstrating Iteration Results**

At the beginning of each iteration, the Agile Project Core Team (A-PCT) commits to a goal agreed with the Product Owner (PrOw). This goal guides the team throughout the implementation of a specific set of Work Items taken from the Work Items List (WIL). By the end of the iteration, the team demonstrates all the implemented Work Items to validate if the iteration goal was achieved.

As described in the previous section, each Work Item exposed contributes to a "growing" and evolving artefact. This process continues throughout the several iterations until a set of functionalities is released according to the strategy defined by the PrOw, as demonstrated in figure 3.12.

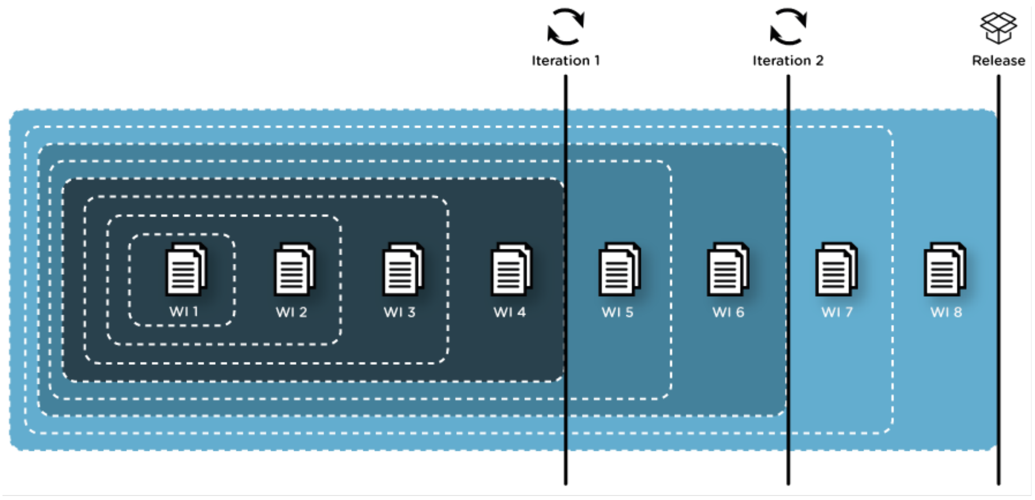


**Fig. 3.12** Solution "growing" towards a release.

While exposing Work Items allows the team to receive quick feedback and helps them to stay on track, it is only when the iteration ends that an evaluation is made to verify if the team was able to reach all the goals they have committed to. This evaluation occurs in the Iteration Review.

The Iteration Review is a prescribed event, where a team aims to collect feedback from the client to ensure that the Work Items are Done (according to their Definition of Done[6]). This event requires a certain degree of formalism. Though much of the iteration content was previously exposed and validated with some degree of confidence, a formal check for each Work Item must be performed.

It's during the Iteration Review that the benefits acquired from the continuous exposure of Work items become visible. This occurs because the Agile Team Members (ATeM) can fully demonstrate the developed Work Items while helping the remaining stakeholders to validate them. Enabling the stakeholders to perform a rigorous and clear validation will result in a positive overall perception of the solution.

The Product Owner (PrOw) provides the primary feedback for the Work Items exposed throughout the iteration. However, during the Iteration Review, the Business Manager (BM), members of the Business Implementation Group (BIG), and other relevant stakeholders should also participate in the evaluation process.

### 3.3.3 Releases

A Release is the result of the incremental process applied throughout several iterations, as the team releases the solution as a viable product (or a Minimum Viable Product) that is ready to be used by the stakeholders.

[6] For more information on Definition of Done, please check the technique "Definition of Done"

Depending on the solution being built, additional levels of formalism and validation may be required. By default, the incremental process, based on the continuous exposure of Work Items and the demonstrated and validated Iteration results, enables a solid Release. Nevertheless, specific requirements may require additional validation steps to ensure that the Release is indeed fully compliant with the agreed Definition of Done.

When developing more complex solutions, the process of releasing may require some additional preparation and some extra activities (training users, fixing a critical bug, etc.). This additional effort may be planned as part of an iteration called a *Transition Iteration*. The main purpose is to ensure a successful solution deployment into the production environment, enabling future exposure to the community of users.

Within a Transition Iteration, the team is essentially focused in guarantying a smooth transition of the solution. For example, the team may decide that no new features are to be further developed and other activities that also add value to the product are carried out instead. An Agile team may also be split into two sub-teams that work together in parallel. One team may work on release activities while the other one is developing new features for a follow-on release, for instance.

Typical activities that occur during a Transition Iteration include:

- fixing a critical or major defect;
- documenting various aspects of the system (e.g. support and user documentation);
- performing additional functional, performance, load, system and integration testing;
- training support personnel and end users;
- updating the Development Plan and/or the Deployment Plan.

One Transition Iteration may not be enough to perform all the necessary activities. A Transition Iteration has some different activities, but it doesn't mean that it is planned differently. It remains an iteration and as such, it should be carried out just like a regular one (proper ceremonies, Iteration Planning with a Work Items List, etc.)

## 3.4 The PM²-Agile Mindsets

PM² promotes a set of mindsets that aim to integrate and complement the four pillars of PM² (Governance, Lifecycle, Processes, and Artefacts). These Mindsets present useful reminders of effective attitudes & behaviours. They help project teams to define and focus on what is (really) important for project success, and help project teams (re)position project goals in the wider organisational context.

The purpose of the Mindsets is to help project teams navigate through the complexities of managing projects and provide a common set of beliefs and values for all project teams.

The eleven PM² Mindsets[7] are presented below as a list of eleven statements for Project Managers and project teams who practice PM²:

1. **Apply PM²** best practices to manage their projects.
2. **Remain mindful** that the methodologies are there to serve projects and not the other way around.
3. Maintain an **Outcomes Orientation** in relation to all projects and project management activities.
4. Are **committed** to delivering project results with **maximum value** rather than just following plans.
5. **Foster** a project culture of clear **communication** and effective **collaboration**.
6. **Assign** Project Roles to the most **appropriate** people for the benefit of the project.
7. **Balance** in the most productive way the project management “P’s” of: **P**roduct, **P**rocess, **P**lan, **P**eople, **P**leasure/**P**ain and **P**erception.
8. **Invest** in developing their technical and behavioural competences to **become better** project contributors.
9. **Involve** project stakeholders in the **organisational change** needed to maximize project benefits.
10. **Share knowledge, actively manage** lessons learned and contribute to the **improvement** of project management within their Organisations.
11. Draw **inspiration** from the PM² Guidelines on Ethics and Professional Virtues (see **PM² Guide Appendix F**)

[7] For more information on the PM² mindsets and the ‘answers’ to the IAQs, please see the relevant section of the PM² Guide.

To remain mindful of the PM² Mindsets, Project Managers (PMs) and project teams that practise PM² should ask themselves the following important Infrequently Asked Questions (IAQs):

- Do we know what we are doing?
- Do we know why we are doing it? Does anyone really care?
- Are the right people involved?
- Do we know who is doing what?
- Deliver at any cost or risk?
- Is this important?
- Is this a task for “them” or for “us”?
- Should I be involved?
- Have we improved?
- Is there life after the project?

Notice that the PM² Mindsets are 100 % aligned with the spirit of Agile as presented in PM²-Agile and this guide. Moreover, these PM² Mindsets are further extended by the PM²-Agile set out below.

**PM²-Agile Mindsets**

The main mindsets of an Agile delivery approach are derived from an interpretation of the Agile Manifesto and can be summarised with the following three points.

- Focus on **solution delivery** (whereas the original manifesto focused on software development, a term that too many people have understood to mean only software development or construction-focused).
- Focus on **all project stakeholders** (whereas the original manifesto focused on ‘customers’, a word that for too many people appears to imply only the ‘paying’ business stakeholders).
- Focus on the **overall organisational ecosystem and its improvement** (whereas the original manifesto focused on development teams).

Drivers.

- There is no such thing as an Agile project. There are only PM²-Agile projects.
- Agile team members are members of the PM² Project Core Team (PCT).
- Agile roles are not people. The roles are fulfilled by people.
- There are clients and users.
- Have a User centricity attitude.
- Making over analysis.
- Be serial in the large, and iterative in the small.
- Speak Agile, but also speak ‘organisation’.
- Organise work around timeboxes, not the other way around.
- Get out of the “deliverables business” and focus on “user experience design”
- Be as Agile as needed and as Agile as possible within the constraints of the organisation.
- ‘Celebrate failure’ as a way to openly discuss mistakes and accelerate learning.

Ongoing Goals.

- Fulfil the project mission.
- Grow team members’ skills.
- Enhance existing infrastructure.
- Improve team’s processes and environment.
- Leverage existing infrastructure.
- Address risk.
- Concentrate on the Flow of value, where the customer defines value.
- Focus on the continual reduction of waste that is impeding flow.
- Transparency, enabling team members to continuously improve the two previous goals.

Rights of everyone.

- To be treated with respect.
- To be provided with suitable working conditions/environment necessary to perform the job.
  - To be informed about the business context.
  - To have decisions taken in a timely manner.
  - To obtain the identified required resources.

    - To respect commitments, both within the team and at the overall project and organisational scope.
- To actively participate in planning and estimation activities, commit to deliver within the quality and organisational standards.
- To have a continuous learning environment, where skills development is perceived as bringing benefits to the individual, the team, and the overall organisation. Mistakes are seen as opportunities to learn and improve.

## 3.5 PM²-Agile Artefacts and Documentation

Agile does not mean 'no documentation', but rather just enough. 'Enough' is defined by both the needs of the project and by the corporate requirements for transparency, traceability, and control.

Documenting the work planned and performed in the Agile layer is critical in increasing transparency and coordination between the different layers of the PM² project organisation (i.e. between the directing, managing and performing layers).

Project documentation (as a source of information and knowledge) becomes increasingly relevant and useful outside the boundaries of the project for decisions at the cross-/multi-project (or 'programme and portfolio') levels. Therefore, Agile teams need to produce documentation that is enterprise-aware (in terms of its content, structure and language), and simultaneously standardised within the organisation and aligned with PM² documentation requirements.

A set of artefacts support the use of PM²-Agile. These artefacts are used to capture and document information regarding the management approach, specific (development) activities, milestones, issues and progress reporting.

These artefacts are grouped into three categories: IT governance artefacts (which are common to all IT projects, regardless of the management approach chosen), Agile (development) specific artefacts and Coordination & Reporting artefacts.

| Artefact Type | Description |
|---|---|
| IT Governance Artefacts | Provide the information requested by the Organisation's IT Governance. These consist of the Business Case, Project Charter, Architecture Overview, and Operational Model documents. |
| Agile (development) Specific Artefacts | Capture information regarding the planning of specific (development) processes, activities, releases, iterations and other milestones. |
| Coordination & Reporting Artefacts | Capture the information needed to coordinate the overall project activities with those undertaken by the Agile Project Core Team (A-PCT), and to ensure that the Project Manager has visibility of the domain-specific activities, issues, milestones and progress. |

The following artefacts support project work based on PM²-Agile.

- Business Case and Project Charter.
- Architecture Overview.
- Operational Model.
- Development Handbook (becomes part of the overall Project Handbook).
- Development Work Plan (becomes part of the overall Project Work Plan).
- PM²-Agile Logs.
- Deployment Plan.
- Test Plans.
- Development Status Report (becomes part of the Project Reports).

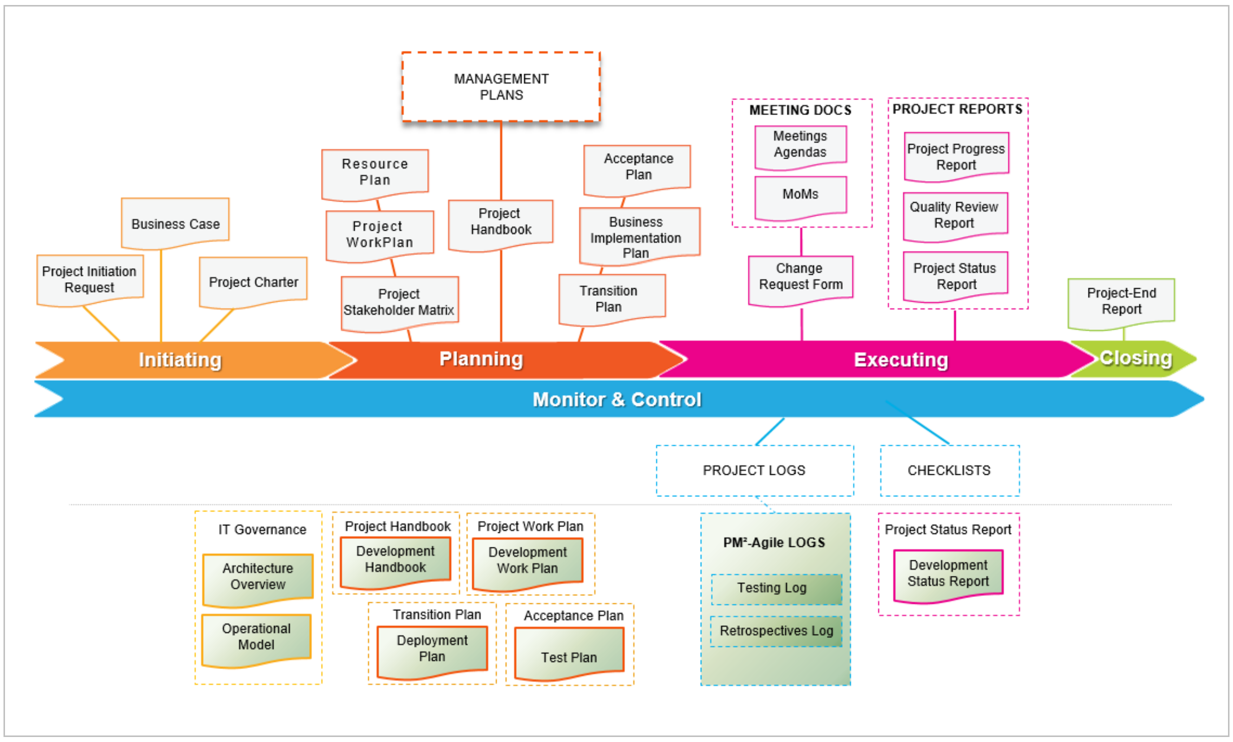


**Fig. 3.13** PM²-Agile (combined) Artefacts Landscape

Note that as per the IT Governance (see section 7-Artefacts), the following documents are mandatory for projects that include information systems as part of the solution.

- Business Case.
- Project Charter.
- Architecture Overview.
- Operational Model.

Depending on the Organisation's structure, the IT Governance may request additional artefacts to the ones mentioned above.

## 3.6 Tailoring & Customisation

PM²-Agile should not be seen as a one-size-fits-all approach. Rather, it should be tailored to address the specific business and project conditions and constraints and allowed to evolve as the project teams learn and discover more (regarding the environment and the solution) through each iteration, release or project.

Tailoring refers to changing specific parts of the methodology, such as process steps, artefacts contents or the distribution of responsibilities amongst the various roles. Organisations do this to adapt the methodology to the specific needs of their structure and culture, and to align with their processes, policies, etc. Tailoring can be also carried out at other levels like a Division, a department or even a smaller sector, and applied to all its projects.

The Development Handbook (see section 7 "Artefacts") documents and describes how and why PM²-Agile has been tailored for a particular project so that it better serves the project needs and goals. This typically includes:

- the rationale for tailoring and deviations;
- information about variations from the standard practices, and why (typically by identifying artefacts or tools and techniques added or removed);
- tailoring of Roles & Responsibilities;
- tool and/or medium and format used for each artefact;
- references to other guidelines and information that the project may use in addition to PM²-Agile and PM²;
- what (development-specific) reviews will be performed and their level of formality;
- procedures and artefacts for reporting progress, performing measurements, managing requirements, managing change requests, etc;
- configuration strategy development-specific assets;

- responsibilities for review and approval (a RASCI table used).

The following list provides typical activities or key considerations when tailoring:

- Involve appropriate stakeholders.
- Identify and create awareness regarding the mandatory policies and procedures to be respected within the organisation for all Agile team members.
- Decide what project-specific content is needed.
- Capture/update glossary of terms (definitions, acronyms, and abbreviations) specific to the task at hand (project-wide glossary or specific-area process).
- Capture references (documents important to the task at hand).
- Decide how and to what extent work will be documented.
- Make sure that the project organisation, project level and Agile-level responsibilities and interfaces are described and clearly understood.
- Decide which tools will be used to create and maintain work products.
- Ensure the right tools and infrastructure are available and the team knows how to use them.
- Be aware of available metrics and reports and decide which ones you will use and how often.
- Determine that team has the training and resources, including personnel that they require.
- Review and agree on the tailoring.

# 4 PM²-Agile Themes

With the introduction of Agile, several project management concepts and practices have to be adjusted because they are now relying on a different set of values and principles. PM²-Agile acknowledges this and organises those concepts and practices in several areas of knowledge called Themes. The goal of this section is to identify them and describe how did those concepts and practices evolved to support this reality.

## 4.1 Lean UX

The Lean start-up model described in section 2.3 "Continuous Improvement & Validated Learning" is a key enabler of the continuous improvement and validated learning activities. It strongly influences Lean UX, another cornerstone of PM²-Agile.

The Lean start-up model supports the Lean UX process in PM²-Agile through:

- removing development waste with UX design;
- harmonising the relationships between the Agile Project Core Team (A-PCT) and the rest of the stakeholders;
- shifting mindsets from assumption to experimentation.

Lean UX helps all participating stakeholders to understand the process and the importance of involving everyone throughout all PM² phases. It focuses on the principle that requirements are assumptions that the Agile Project Core Team (A-PCT) needs to validate before start developing. Development takes time and effort to reach the level of marketable product required to gain learning and bring business value. Therefore, rapid prototypes establish the appropriate learning that enables the team to move forward. The motto is that no-code prototypes foster a common understanding, continuous improvement and validated learning among team members and stakeholders.

Design blocks provide the basis to challenge and validate assumptions as well as help developing MVPs.

Lean UX ensures that all key elements described in section 2.3 'Continuous Improvement & Validated Learning' are taken into consideration when building an MVP.

The 4 steps of the Lean UX process have been adapted and integrated seamlessly in the PM²-Agile lifecycle, with a key focus on the Initiating and the Executing phases. This integration is further detailed in the following sections:

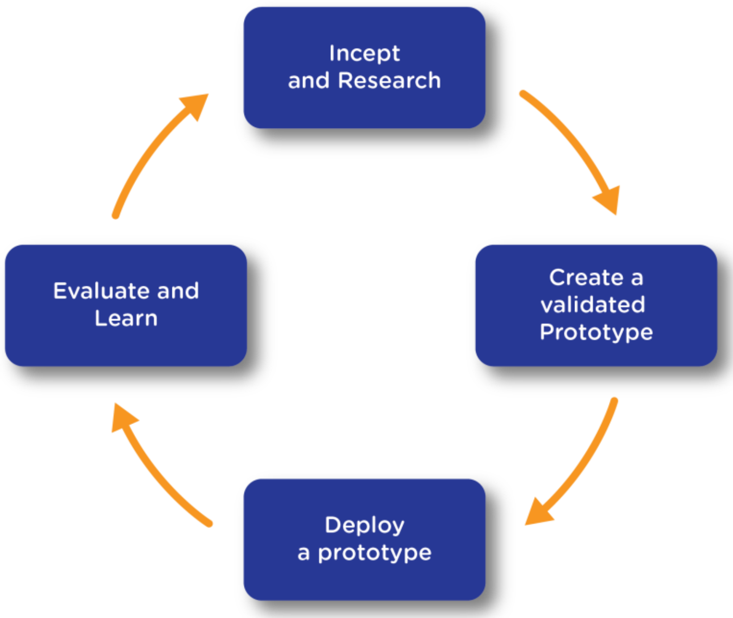


**Fig. 4.1** The PM²-Agile Lean UX process[8]

[8] Adapted from: *Lean UX – Designing Great Products Agile Teams*: Jeff Gothelf and Josh Seiden – O'Reilly 2016

**Incept assumptions and research**

This step plays a pivotal role both in the Initiating phase of PM² (as the foundation to help defining requirements) and in the Executing phase (at operational level as Design Blocks).

During the Initiating phase, co-creation sessions with the stakeholders will take place to give them the opportunity to ideate and express their expectations, organisational goals and create common understanding. This inception of ideas, assumptions, will drive the future requirements and consider several alternative solutions by analysing their strengths, weaknesses, opportunities and threats (SWOT analysis), as described in the Business Case. The team can then visualise the solution's ecosystem and proceed to move forward with a first MVP if the project is approved.

During the Executing phase, Design Blocks are also used in every iteration to revisit the hypotheses of the relevant features and (re)ensure common understanding. As changes in the solution context may occur and learning from previous iterations surfaces, this exercise plays a very important role in keeping the expectations aligned while ensuring common understanding.

**Create a Validated Prototype**

Lean UX suggests that rapid prototypes are essential because there is barely development effort that might delay validated learning, and thus, create waste. During the Executing phase (supported by the iterative nature of PM²-Agile), the Agile Project Core Team (A-PCT) reviews assumptions and hypotheses and then uses Design Blocks to develop and validate low-fidelity prototypes with the stakeholders.

When the team and the stakeholders reach a common agreement, UI designers develop high-fidelity prototypes and take them back to the stakeholders and users for validation. Once this occurs, the prototype is ready to be deployed.

**Deploy a Prototype**

In this third step, and following the common understanding and agreement between the business stakeholders and the Agile Project Core Team (A-PCT), the prototyped experience is deployed. Note that the act of deploying doesn’t necessarily imply a technical deployment. The goal is to freeze an experiment that is ready to be taken to the next step and be used as a validated learning tool.

**Evaluate and Learn**

After deploying the high-fidelity prototype, real and objective evidence becomes available as a mean to assist the entire team validating the hypotheses that were formulated. This evidence generates objective learning that is used to support decision-making, including to pivot or persevere over a feature, a set of features or even the entire solution. When persevering, the preparatory work will allow Agile Team Members (ATeMs) to start building the specific elements of the solution as part of their work in the following iterations.

Agile Team Members (ATeMs) facilitate usability testing, user and stakeholder interviews. These sessions are open to everyone and allow the team to test if the hypotheses assumed are aligned with business stakeholders and users' expectations. This activity can either be an informal feedback session or a key ceremony in PM²-Agile, such as an Iteration Review or a Release Planning meeting.

The Lean UX mindset is focused on co-creation and supports the PM²-Agile Themes. By bringing together Agile teams and stakeholders, they continuously engage and progressively validate the digital solutions delivery in terms of planning, coordination and reporting, requirements, etc.

## 4.2 Planning

One of the common misperceptions about Agile relates to planning and originates from one of the Agile core values: 'Responding to change over following a plan'. Note that this refers to following a plan, and not to planning. The amount of planning effort in Agile involves more effort in comparison to traditionally managed projects. From maintaining the Work Items List and estimating work items effort, to Stand-up Meetings and iteration planning, the success of Agile projects depends explicitly on effective and efficient planning.

There are several differences between Agile and traditional project planning. For example, the intensity of the planning activities in Agile is spread throughout the whole project lifecycle.

There are also differences in the detail, granularity and format of the plans as well as the various key players involved with planning activities.

Planning plays an important role in the integration and coordination of the efforts of people. Different levels of involvement in the project, time constraints, planning horizons and responsibilities must be considered. The flexibility of the planning methods should allow the fulfilment of the project coordination role for the delivery of ideal digital solutions. For certain elements of the plans (including presentation and granularity) to be accessible and transparent to all team members and stakeholders, agreements should be established. Without these agreements, it is difficult to commit resources or synchronise various parallel work streams. Lack of visibility and coordination can result in disconnects, inefficiencies, and can consequently increase tension and friction throughout the organisation.

For example, there is a controversy about the need (and timing) for planning outputs with concrete schedules, definite budget estimates, and breakdown of activities based on agreed and complete requirements. This is because Agile teams (justifiably) do not focus on long-term detailed planning; they are not inclined to create detailed plans, traditional timelines and cost estimates. Instead, resource requirements and timelines are measured differently, using iterations, story points and velocity for their respective work. Releases are based on a 'cut-off' date as opposed to well-defined requirements and production is founded on increments integrating feedback from the client and other stakeholders. However, there is a need to manage the expectations of the Managing and Directing stakeholders who usually prefer to receive complete plans before the project starts. By involving and educating these stakeholders in transparent Agile planning and progress tracking processes, the efficacy of the Agile process can be demonstrated.

PM²-Agile proposes sufficient upfront planning and documentation that complies with IT Governance and project management requirements. This process guides subsequent iterative and incremental planning and development.

PM²-Agile promotes the following principles for planning:

- The planning cycles (and scope) should be decided per the cycle they are part of and the nature of the project.
- Everybody is involved in planning.
- Planning is more important than plans: “Do the planning, throw away the Plan”.

## 4.3 Coordination and Reporting

Agile practices have developed specific terms, rituals, formats and metrics to communicate efficiently and coordinate team efforts. However, an Agile Project Core Team (A-PCT) does not work in isolation. There might be other non-Agile teams in the project accompanied by various groups of stakeholders with specific informational needs and preferences. PM²-Agile addresses these needs by adjusting the types of information, formats, granularity, language and frequency of communication accordingly.

The coordination between the A-PCT and other groups of stakeholders can be on different themes. The following themes are relevant for most types of IT projects.

- **Planning** - The A-PCT has to provide a Release plan that is easy to understand and integrates with the overall project work plan. In addition, some content from the iteration plans can also be used outside the A-PCT.
- **Reporting** - The specific progress metrics and the frequency of reporting of the A-PCT should be aligned with the accepted metrics for the other teams and stakeholders (e.g. from management layers).
- **Architecture** - Architecture artefacts should be created following organisation standards so that impact assessment, reusability, interoperability and other types of analysis can be executed without additional overheads.
- **Deployment and Transition** - The Project Core Team (PCT) and the Agile Project Core Team (A-PCT) should work with the IT operations team. Transparency enables collaboration which supports the integration of processes, such as Software Configuration Management.
- **Risk** - The A-PCTs may capture both technical and other projects risks following the PM² Risk Management Process and using the recommended artefacts (e.g. Risk Log).

The **external coordination** between the A-PCT and other project layers and stakeholders is facilitated through regular project meetings and reports. External communication events include:

- Project follow-up meetings
- Project Status Reports
- Demos and Deliverables Acceptance events

Issues may occur in external coordination relating to the type and granularity of the planning and reporting unit. There should be three main groups of reporting and planning.

- Delivery units (e.g. work packages, deliveries).
- Functional units (e.g. use cases, business processes, capabilities, features).
- Work units (e.g. work items like user stories, enabler stories, bugs).

Wherever possible and applicable, these units should be linked to each other. For example, the smaller work items (stories) should be related and grouped into a functional unit (e.g., a feature). Consequently, a group of Features (functional units) should be linked to a specific delivery unit (for instance, a Delivery). This traceability will allow more effective external coordination by providing the stakeholders with a clear vision of current project execution and delivery.

**Internal coordination** within an A-PCT is supported by open, frequent, efficient and face-to-face communication. Figure 5.1 shows several Internal communication ceremonies that include:

- Daily Stand-up
- Iteration Planning
- Iteration Review
- Iteration Retrospective

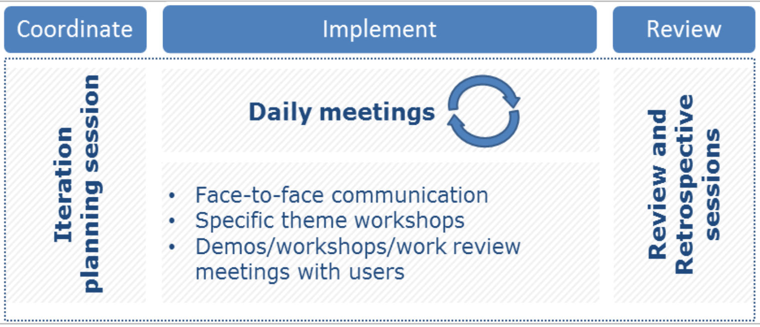

Coordinate
Implement
Review
Iteration planning session
Daily meetings
Face-to-face communication
Specific theme workshops
Demos/workshops/work review meetings with users
Review and Retrospective sessions

**Fig. 4.2** Communication

## 4.4 Requirements

In general, requirements represent a description of product or service features that are required to satisfy the stakeholders' needs.

If requirements could be agreed and fixed early in the project life cycle, that would allow the team to effectively:

- document exactly what will be delivered;
- estimate the human-resource requirements;
- develop the project budget;
- build a complete project work schedule so resources can be committed.

If both the requestor and the provider are certain that the solution is completely described by the requirements specification, then the traditional upfront planning approach can be used. However, when the requirements are not entirely known or if the rate of changes is high, Agile and Lean UX approaches yield better results. Agile project organisation, if managed well, creates the conditions for requestor and provider co-evolution by being regularly exposed to the interpretation of their agreements. Therefore, the next instalment of requirements definitions can be adapted to the lessons learnt.

However, the identification of requirements is itself prone to some strategic risks.

- Requirements may be derived from some knowledge of possible solutions. As such, they represent the 'how', rather than the 'what', thus limiting the possibility to discover the most appropriate solution.
- Requirements represent investment items. Meeting and maintaining each new requirement is a regular additional expense. Furthermore, these IT investments can represent duplications if requirements are for building or buying capabilities that are already available. To minimise this risk, it is important to analyse the available capabilities, components, and services and their feasibility for reuse.

- An important first step is to identify requirements by clarifying the problem to be solved or the needs to be addressed. Without this effort, problem-solving opportunities may be missed or the consequences of implementation on the entire enterprise architecture may be underestimated. Therefore, it is important to fully document the problem or need in the PM² Business Case before gathering requirements. Then once the solution architecture is clearer, it's then necessary to evaluate the impact of implementing it.

In order to ensure proper requirements' management, implementation and monitoring, PM²-Agile considers two very important dimensions to describe a solution's behaviour:

- **Abstraction Level** –There are several different levels of abstraction to identify requirements. PM²-Agile acknowledges the key role of Features and Stories to describe the solution's behaviour.
- **Implementation Perspective** – The implementation perspective of a solution cannot be restricted to a business point of view. To have a solution built, it's of paramount importance acknowledging and understanding all the underlying work that needs to be done to support the implementation of the business features. That is the key role of PM²-Agile Business and Enabler Items.

**Features and Stories[9],**

A Feature is a proposed service, within the context of a solution, that exists to address the needs of the business, the customer and the user. The PM² Project Charter foresees a more detailed description of the proposed solution through the identification and description of the features and the corresponding needs they will target.

Features are described in the Project Charter with a high level of abstraction to ensure all layers in the project's organisation clearly understand the scope of the solution. When described, features should provide information about their context and the benefit hypothesis that must be verified upon delivery.

To execute a predefined Release strategy in the most effective way, features must be prioritised. This activity should involve key stakeholders such as the Business Manager (BM), the Project Owner (PO) and the Business Implementation Group (BIG).

To help with prioritisation activities, PM²-Agile strongly recommends the use of combined elements like Business Value and the "Size" of a feature. Features that deliver the highest business value with the shortest turn-around time will be delivered first. Additionally, projects with a higher risk level should also take into account the risk reduction factor of each feature. Features with higher risk factor should have higher priority.

A feature is developed and delivered in the context of a release throughout several iterations. Because the Agile Project Core Team's (A-PCT) cadence is based on iterations (smaller cycles), the features (functional units) must be decomposed in a set of stories (working units) so that they can be planned and implemented within each iteration. By evaluating the output of each iteration in terms of completed stories, one can easily measure the progress of each feature or release and make any necessary adjustments when needed.

A story is a piece of functionality that is written using a language focused on the user. Implemented by the Agile Project Core Team (A-PCT) within an iteration, the story represents a small, vertical slice of system behaviour that supports incremental development.

Stories provide just enough information for both business and technical people to understand the intent.

Details are deferred until the story is ready to be implemented. Through acceptance criteria and acceptance tests, stories become more specific, helping to ensure system quality.

The main consumer of stories is the Agile Project Core Team (A-PCT) though other stakeholders like the Business Manager (BM) or the User Representatives in the Business Implementation Group (BIG) can also use them.

Stories are a simple and straightforward way to identify and keep track of requirements. Because they are brief, the overhead is low for creating and updating them when requirements change. Focusing on delivering value, stories are a good basis for prioritising and implementing requirements.

Stories in PM²-Agile are referred as Work Units or Work Items and collected in a prioritised Work Items List[10].

[9] For more information on Features and Stories, please check the "Features and Stories" tool/technique

[10] For more information on how to breakdown a Story, please check the "User Stories Breakdown" technique

**Business and Enabler Items**

Implementing a solution focuses on satisfying the needs identified by the stakeholders. When defining the solution scope and identifying the desired features, the adopted perspective is almost exclusively focused on the business. However, when defining and developing a solution, one must look beyond this business perspective and focus also on the non-business effort that will enable the implementation of the required features. In PM²-Agile, this non-business focused effort is referred as Enablement Work and it includes Architecture, Infrastructure, evaluating options, legal compliance, continuous improvement and other non-functional requirements.

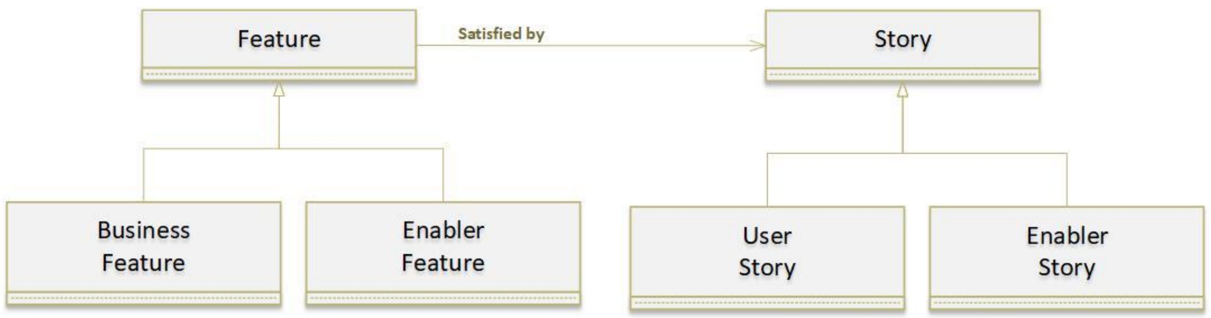


**Fig. 4.3** Features and Stories relationship

A Feature that has a business perspective is known as a Business Feature while a Feature that has an enablement perspective is known as an Enabler Feature. Because enablement elements compose the solution to be planned and built, they must be always visible.

Although the business and enablement concepts were mentioned as related to features, they also apply to stories. A business perspective story is known as a User Story while an enablement perspective story is known as an Enabler Story.

Due to their importance, enabler items must be considered when prioritising the Work Items List (WIL). The Architecture Owner (ArOw) plays a very important role, helping the Product Owner (PrOw) finding the right balance between business and enabler work. Putting the focus exclusively on building business items may yield good initial results. However, the Agile Project Core Team (A-PCT) will struggle to deliver once the lack of a solid architecture becomes obvious and dramatically impacts their velocity. Putting the focus exclusively on building enabler items will produce a robust architecture solution but without delivering the functionalities that users are expecting.

For more information about Features and Stories, please refer to the PM²-Agile Tools & Techniques guide.

## 4.5 Estimation and Prioritisation

Agile estimating differs from 'traditional' upfront detailed estimates as it incorporates the notion of *just enough upfront detail*. It balances the effort that should be put in estimating with the amount of information available and the expectations or needs regarding the accuracy of the estimations.

PM²-Agile estimation is based on three main concepts.

- **Relative Estimate** gives a high-level estimate for the size of a work item, typically measured using a neutral unit such as points.
- **Velocity** measures the number of points the Agile Project Core Team (A-PCT) can deliver within an iteration.
- **Absolute Estimate** translates the size (measured in points) of a work item into an objective, detailed estimate of effort, typically using the units of Actual Hours. The estimation of effort indicates how much time will take the team to complete a specific work item.

Relative Estimates are based on the fact that humans are better at making relative estimations[11]. Instead of person-hours or person-days to determine the size of a Work Item, PM²-Agile uses points to estimate Work Items in relation with other Work Items.

[11] For more information on relative estimation, please check the "Planning Poker" tool/technique

Relative estimates tend to be more accurate as the project progresses and both the problem and the solution becomes clearer to the project team.

Team velocity is important to track in terms of planning regular iterations and measuring team performance. Detailed guidance regarding the point-based Agile estimating technique is provided in the Agile Tools and Techniques publication.

In addition to estimation, Work Items are prioritised according to the delivery value and their contribution toward achieving project goals. Agreeing on and negotiating the priority of each work item considering the team's estimates, enables contingency management for these projects.

The MoSCoW technique for example ('Must have', 'Should have', 'Could have', and 'Won't have' this time) empowers the team to make decisions collaboratively and consciously on what constitutes the 'minimum usable subset' or 'minimum viable product'. Some methodologies suggest that 60% of the total effort should be assigned to the 'Must have' requirements, 20% to the 'Should have' requirements, and 20% on the 'Could have' requirements. Thus, in cases when 'Should have' and the 'Could have' Work Items cannot be tackled in the specified time frame, contingency is 'applied' by still delivering what were agreed to be the most relevant features of the information system – the 'Must have' requirements.

Both estimating and prioritising are Work Items that should happen continuously throughout the project. Therefore, as the project progresses, the plan on how to achieve the desired goals can be better adapted.

## 4.6 Risks

Agile Risk Management involves removing obstacles that impede the work progress of the project team members. These obstacles, also known as impediments or roadblocks, are any situation or event that prevent or block the progress of work during an iteration. Removing these obstacles reduces the overall risk of a PM²-Agile Project.

It's often mentioned that PM²-Agile projects are less risky by nature. This is true and the reason goes back to the foundations of the Agile Manifesto. PM² Project Risk Management activities are also used in PM²-Agile and performed throughout every iteration.

The following Agile Manifesto principles represent a clear example of how PM²-Agile is less risky by nature.

- **Collocated teams** - Bringing the project team and the client together, preferably in the same physical area, leads to better communication and less wasted time.
- **Simplicity** - Maximize the amount of work not done: The team should only concentrate on the minimum amount of work that needs to be performed. Wasteful processes should be eliminated and the practice of Gold banned. Removing waste is a risk management practice that forms the foundation of PM²-Agile.
- **Regular Reflection** - By continuously improving processes throughout the project, risk is exponentially reduced. Process improvements should be immediately implemented in the following iterations, ensuring that these same risks do not occur again in the project.

In PM²-Agile, there are two types of Risk Management approaches that help materializing how Agile is perfectly aligned with the PM² Risk Management Process. Those are:

- **Implicit** - It's what is intrinsic to the Agile process, making risk management coming out "naturally" from the natural iterative planning and review process and activities.
- **Explicit** - Refers to implementing risk management strategies in order to identify, assess and mitigate project risk by using specific agile risk management tools and techniques.

To give a better idea of the concepts of Implicit and Explicit Risk management in PM²-Agile projects, below there's an explanation of how each of these concepts apply to each of the four steps foreseen in PM$^2$ to manage risk.

- **Identify Risks** - In PM²-Agile, the whole team implicitly identifies risks during ceremonies like Iteration Planning, Daily Stand-up, Iteration Retrospective, etc, recording their results in white boards and other tools. If the team is using an explicit strategy, additional topics can be added to a planning meeting agenda to explicitly identify and prioritize risks. The result will influence the work that is being planned for that particular iteration. Additionally, risks can be continuously identified during the Daily stand-up meeting, either as obstacles (implicit strategy) or as risks (explicit strategy).
- **Risk Assessment** - Projects using PM²-Agile generally perform only "qualitative analysis" at the expense of "quantitative analysis" as Agile short development cycles and constant reviews make this

feasible and effective. Qualitative elements like Risk Likelihood and Risk Impact, based on a relative scale of one to five, make Risk Assessment a team exercise and provide consolidated information to the risk information radiators.

- **Develop a Risk Response Strategy** - In PM²-Agile, the entire team is implicitly involved in risk response planning in every project iteration, as they all participate in the planning meetings, daily stand-ups, retrospectives, etc. When an explicit approach is required to manage risks, tools and techniques such as risk-based spikes or risk Boards help to develop immediate risk response and implementation strategies.
- **Control Risk Response activities** - Risk response is a continuous and ongoing process described In PM². Due to its iterative nature, PM²-Agile has these monitoring and controlling processes implicit. Daily stand-up meetings allow risks to be monitored regularly by exposing potential threats and obstacles.
- In PM²-Agile, risk reassessment occurs at least by the end of the iteration, during the Iteration retrospective, where previous risks and concerns are revisited to determine changes to be implemented in future iterations.

## 4.7 Quality

PM²-Agile references two different but complementary quality types – the quality of the development process and the quality of the outputs of the process.

As PM² states, project quality management aims to ensure that the project will meet the expected results in the most efficient way and that the deliverables will be accepted by the relevant stakeholders. Quality is the result of collaborative work, common understanding and empathy between all the stakeholders. It is an indicator of success, when the end user perceives the outputs of the process as useful and intuitive. In PM²-Agile, the quality of the development process is managed by tailoring the approach that best fits the project environment and by frequently involving the whole Agile Project Core Team (A-PCT), the Business Manager (BM) and the User Representatives during each timeboxed cycle of work. Altogether, they must focus exclusively on generating outcomes that will be perceived as valuable by the Project Owner's organisation.

To achieve fit-for-purpose deliveries, close collaboration between the Project Owner (PO) and the Solution Provider (SP) organisations along with the use of Agile tools and techniques is necessary. Timeboxed cycles of work and frequent user demos provide the time and space for collaborative improvement. This allows the Agile Project Core Team (A-PCT) to assess if the solution is heading in the desired direction by providing maximum value to the business stakeholders.

## 4.8 Evolution & Change

The Evolution and Change dimension of a project highlights the need to recognise and to welcome the natural evolution of the project scope. As a project progresses, both the provider and requestor organisations gain a better understanding of the problem to be tackled and how to address it.

As PM²-Agile enhances and extends the PM² methodology, changes that seriously impact the development aspect of the project should be managed within the overall project change management process. Agile Team members (ATeM) should also participate to ensure proper understanding of the change requests.

From a specific PM²-Agile perspective, it is important to ensure that change requests are translated into the corresponding Work Items that will populate the Work Items List (WIL). Those Work Items will be prioritised and later implemented in the scope of one or more iterations.

PM²-Agile addresses the natural and expected evolution and change during the project through **Release Planning** and **Iterative and Incremental Development** practices.

The **Release Planning** practice focusses on the high-level (macro) planning of the complete known project scope. Each planned release includes all the work items to be released and describes when they will be implemented across the several iterations. It is based on the notion of creating a project coarse-grained plan and use it to guide the development of fine-grained plans for each iteration. As such, it's clear that a properly estimated and prioritized Work Items List (WIL) is critical to ensure proper release planning.

Release Planning is a continuous effort that should reflect the team's growing knowledge of the solution. This effort improves the accuracy of project planning, delivering updated Release Plans that clearly show what the team expects to deliver in each planned release.

When adopting Release Planning, the Agile Project Core Team (A-PCT) can better adjust itself as a response to the need of increasing manpower. Also, Release Planning makes it visible to the PrOw how the solution is evolving, giving him enough time to react in case things don't go as initially planned.

The Release Plan is the main output of the Release Planning activity. Part of the Development Workplan, It's a key element for managing the evolution and change within the development aspect of a project and to communicate progress and trends to senior management. For more information about the Release Plan please check section 7.2.2.2, "Release Plan".

The **Iterative and Incremental Development** practices describe how to create a solution in increments while repeating/refining a cycle over and over again to collect feedback and adjust accordingly.

**Incremental Development** is a pattern used by most of the projects. It is based on the idea that a system is built in a series of increments and each increment adds a subset of the final system's functionality. The system grows to become more and more complete over the course of the project's iterations. The alternative to Incremental Development would be to develop the entire system with a unique, big-bang integration at the end.

PM²-Agile relies on Incremental development as a great tool to facilitate planning as it breaks up the solution and organises it according to a specific strategy. This is the foundation for a solid Release Planning activity.

**Iterative Development** is based on the idea that time must be put aside to review and improve parts of the system based on the feedback from the stakeholders and the self-learning process of the team. The alternative strategy to Iterative Development would be to get everything right the first time.

In PM²-Agile all iterations are timeboxed and with the same duration. This is important for three reasons:

- Establishes the 'heartbeat' of the project.
- Reduces overall uncertainty (there are many things that cannot be predicted, but we can control when an iteration ends).
- Helps understanding the project team's performance.

Working in small and regular Iterations is a great way to collect valuable feedback. **Regular** ensures that the team doesn't go off limits and **small** makes is easier to apply the necessary changes resulting from the feedback.

Small and regular Iterations also support the teams in postponing design decisions to the last responsible moment (referred to as Set Based Design). Those iterations are very important learning cycles that help the team narrowing down their options as they progress, instead of taking all those decisions right from the beginning, when there's not enough information available.

## 4.9 Architecture

When implemented, specific software architecture decisions made in every IT project, complement the overall enterprise architecture. Just like with other Agile practices, PM²-Agile applies this same principle to solution architecture, ensuring balance between autonomy and coherence.

The Agile Project Core Team (A-PCT) should work in an environment that supports original design decisions and innovation. To achieve long-term efficiency (e.g. by reuse of services and components), interoperability and flexibility, the team should follow the enterprise architecture principles and standards.

As defined by an IT Governance Body and supported by an Enterprise Architecture Office, PM²-Agile uses two key artefacts as a standardised way to document the solution architecture and the related deployment aspects.

- **Architecture Overview** - Projects that include an information system development aspect should use this artefact. It conveys the governing ideas (compliance) and illustrates the essential nature of the proposed architecture and its major building blocks.

- **Operational Model** - Describes the operational aspect of the architecture, focusing on the IT system infrastructure. The Operational Model is highly recommended for information systems that are hosted in a corporate data centre.

These two artefacts provide an established way of documenting certain solution architecture decisions. However, they do not ensure a structured and comparable way of documenting that allows an impact or compliance analysis. One of the tasks of an Architecture Office is to improve and harmonise the applied Enterprise Architecture methods and techniques.

As the solution becomes clearer and the software architecture and infrastructure requirements evolve, both artefacts should be kept up to date. Depending on the progress of the project and the impact of the changes, the artefacts should be periodically re-submitted to the IT Governance body.

Depending on the organisation, the importance of upfront architecture design and documenting decisions, patterns, etc. may vary. If a Reference Architecture already exists, architectural mechanisms, patterns, and styles are expected to be used has they have already been evaluated, chosen and standardised by the organisation.

PM²-Agile addresses software design decisions through the practice of Evolutionary Architecture. This practice describes how to incrementally build and improve the software architecture while uncovering and addressing architectural issues found during software development. This reduces technical risks without significant upfront architectural effort. This practice:

- improves quality and productivity by reducing the need to make time-consuming, error-prone fixes to late-detected problems that result from architectural flaws. This is possible because the architecture is validated early, allowing key architectural problems to be corrected before most of development work is done;
- reduces time to market by focusing on reusing existing components. It improves the consistency and maintainability of the system by incorporating lessons learnt from development back into the architecture.
- increases predictability by identifying and implementing the highest-risk technical areas first. It improves the team's responsiveness to change by shortening the architectural cycle and minimising time wasted in architectural rework when changes arise.

The key principles of Evolutionary Architecture are:

- Perform architecture work 'just in time' for all other work.
- Document key architectural decisions and outstanding issues.
- Implement and test key capabilities as a way to address architectural issues.

It's a common practice to perform high level architectural modelling in early stages of the project life cycle. This helps the team and other stakeholders agreeing on the technical strategy to follow during the project. It also improves productivity, reduces technical risks and development time, improves communication and team organisation and enables scaling Agile software development.

When adopting Evolutionary Architecture, the baseline Architecture Overview and Operational Model will evolve. Changes to these artefacts should be managed and documented within the Project Core Team (PCT), ensuring both artefacts are up-to-date and all the impacted stakeholders are informed of the main changes.

## 4.10 Compliance & Security

The PM²-Agile model is enterprise-aware and considers the Organisation's policies and communications related to information systems when addressing the following aspects.

- Document management.
- Data protection.
- Security.

**Document Management**

When building a solution that incorporates a document management aspect, regardless of the medium, it is important to consider the organisation's defined procedures when implementing it.

PM²-Agile recommends creating a checklist based on the defined procedures to verify if document management considerations are applicable to the project. This checklist should be used when outlining the Architecture Overview.

Another key aspect of the document management approach is to guarantee as much interoperability as possible between systems and users.

**Data Protection**

PM²-Agile recommends that guidance on data protection is defined as part of the software engineering practices and must be aligned with the specific legislation. By default, PM²-Agile focus on the General Data Protection Regulation (Regulation (EU) 2016/679), which contains provisions and requirements related to the processing of personal data of individuals.

When developing a solution, specific data processing requirements must be documented. Addressing data protection issues also requires close collaboration between several roles to ensure that the need to process personal data is properly managed. The Architecture Overview must reflect all these considerations when proposing the high-level architecture's strategy.

**Security**

Because Security is a critical aspect in any organisation, PM²-Agile recommends defining a specific practice to address it when building a solution. This practice should include a Security Plan that documents all security-related activities and lists all the required actions to ensure that the system is developed in a secure environment and contains the appropriate security features.

The Security Plan should define the security requirements and provide a step-by-step management plan to meet those requirements. It should also be part of an encompassing Security Certification Statement whose purpose is to test and evaluate the technical features of the system, review the related administrative, personnel, and physical safeguards anticipated for the system's environment, and to provide senior management's formal acceptance of all residual risks.

Additional elements to add in a Security Certification Statement are the Security Test Plan and Report, and the Security Risk Assessment.

## 4.11 Development

The core of PM²-Agile lies on the development of information systems that are part of the project's solution. In this context, development activities such as designing, developing, building, testing, turn the specifications turn into something tangible and allow the system to take shape.

The main goal of the development activities is to ensure that the requirements are properly designed, implemented and made available through an iterative and incremental approach. The requirements should also comply with architectural, infrastructural and governance constraints and deliver the value expected by the business stakeholders.

To help providing the necessary 'agility' to the development aspect, PM²-Agile recommends three different but complementary practices: Continuous Integration, Evolutionary Design and Test-Driven Development (TDD).

With **Continuous Integration**, team members integrate their work frequently (at least daily).

The effort required to integrate a system increases exponentially with time. Therefore, by integrating the system frequently, the overall integration effort is reduced because integration issues are identified and resolved earlier. This results in a higher-quality product and more predictable delivery schedules and activities.

The essence of Continuous Integration can be described by the following activities.

- After making changes in their validated workspaces, developers will perform unit-tests before making them available to the team.
- Change from all developers are merged in an integration workspace and tested frequently (at least daily but ideally, any time a new change set is available).

The first activity ensures that changes are made in a configuration that is known to be good and tested before making them available. The second activity identifies integration issues early so that they can be corrected while the change is still fresh in the developer's mind.

The **Evolutionary Design** practice is based on the assumption that design evolves over time. It minimises the need for documentation while still providing guidance for making design decisions and communicating them. During each round of design, developers add, refine, and refactor the solution. This can be summarised by the following.

- Understand new requirement details
- Identify design elements
- Determine how elements collaborate to realise the scenario
- Refine design decisions
- Design internal elements
- Communicate the design
- Understand the architecture

- Evaluate the design

Test-Driven Development (TDD) describes an approach to development in which test cases are created before the actual code. These tests work as guardrails since the code being developed must pass these tests in order to be considered acceptable.

Test-Driven Development reduces delivery time by decreasing the time required to integrate and stabilise builds. It improves productivity by finding and fixing errors close to the point when they are introduced.

TDD also increases the overall quality of the software by guaranteeing that new developed code has been tested and existing code has been regression tested, prior to check-in.

The practice of Test-Driven Development[12] changes the way developers think. Developer tests are not written as an afterthought but instead, as part of the everyday way of building software.

## 4.12 Software Configuration Management

When developing a solution, hundreds of artefacts like documents, code files, scripts, deliverables, etc, are created and need to be managed. In PM²-Agile, Software Configuration Management provides key activities that will consistently control the changes to the configuration and maintain its integrity and traceability.

Software Configuration Management has four primary objectives.

- **Identify and manage all items that are part of configuration management** – The Agile Project Core Team (A-PCT) must identify all the items (classes, libraries, test scripts, deployment files, etc) they need to keep track of and manage them with the help of specific tools.
- **Simplify the build of different versions of the solution** – Solution development is an incremental process where different versions are regularly being demonstrated and delivered. As such, it's of paramount importance to easily build and deliver a specific version of the solution.
- **Ensure quality is maintained as configuration evolves** – As the configuration continuously evolves, it's important to ensure that the quality of the solution is maintained. This is done by keeping track of the configuration items, their history of changes and the ability to see the impact of those changes.
- **Increase Productivity by reducing mistakes** – Using the right set of tools, Software Configuration Management can be automated to the point where human intervention becomes residual. By dramatically reducing human intervention and consequently, mistakes, productivity will increase.

As a process, Software Configuration Management has a set of steps that PM²-Agile acknowledges as key enablers of the objectives aforementioned.

- **Identification** – Uniquely identify each configuration item to be added to the configuration management repository. Using an object oriented approach is usually a good practice, since this will also facilitate aggregation in similar types of objects (documents, scripts, etc). Properties like name, version, type of item, resources needed, etc are some of the most commonly used.
- **Change Control** – The Change Control is already foreseen as a PM² process that defines several activities related to project changes, including documenting, assessing, approving,
- **Version Control** – setup a set of procedures and tools to enable the creation and management of the multiple occurrences of each object in the configuration management repository. For software development, it should be possible for an Agile Team Member (ATeM) to collect all relevant configuration objects and build a specific version of the solution.
- **Configuration Auditing** – This is clearly a quality assurance activity. The goal is to ensure that quality is maintained as changes are made. The extent of this activity will depend on the context of the project. Typically, it addresses questions like verifying if the change was properly highlighted and documented and the author identified or if the configuration management procedures for noting, recording and reporting the change have been properly followed. Configuration Auditing will help ensuring that the correct configuration objects have been incorporated on a specific build.
- **Reporting** – Provides information to anyone within the project (and organisation) who requires information regarding the changes occurred. Reporting should detail the change initiator, the time/date of the change, the scope of the change and other relevant information. Triggered when

[12] For more information on Test-Driven Development, please check the "Test-Driven Development" tool/technique

a configuration audit is conducted or when new or updated information is assigned to a configuration object, reporting should be aligned with the PM² Communications Management Plan.

As a configuration item moves through the process, the actions implied in one of the steps may not be applicable. As an example, a change in a configuration script may not need the actions involved in the Change Control step, as a change request is probably not needed.

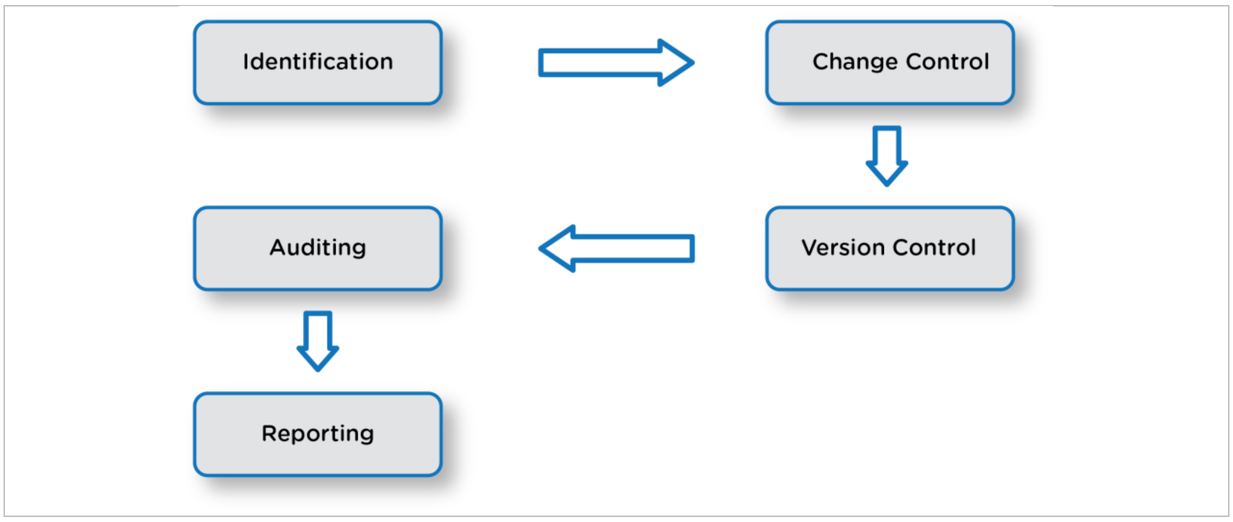


**Fig. 4.4** Steps for Software Configuration Management

Any relevant development-specific considerations regarding the software Configuration Management (e.g. tools to use, templates to apply), should be documented in alignment with the PM² Quality Management Plan.

The following are typical activities and key considerations when tailoring the Configuration Management process.

- Assign each Configuration Item to a specific version of the solution.
- Maintain a record of each Configuration Item (name, creation date, last update, version, etc) for audit purposes.
- Describe conventions for naming and versioning project and product work products in compliance with corporate regulations, if applicable.
- Adopt a standard product directory structure to place artefacts under version control.
- Synchronisation control ensures parallel changes will not override each other.
- Analyse issues involved in setting up the Configuration Management environment, which include:
    - Anticipated size of product data.
    - Distribution of the product team.
    - Physical location of servers and client machines.

Adopt a standard template for the project's Configuration Management system that establishes appropriate streams, roles and responsibilities.

Ensure a Configuration Management capability for managing change requests, including enhancements and defects generated by the team and external stakeholders.

Be aware and comply with policies for project media storage and release.

Document how software developed by vendors and subcontractors is incorporated into version control.

## 4.13 Testing

Agile testing activities should be iterative and incremental. Applying the strategy of 'test early and test often' allows risks to be identified and addressed early in the project's life cycle. It is recommended that the testing activities occur in each iteration, beginning with the earliest builds of the system. Depending on the frequency of new builds, one iteration can have several test cycles.

Testing activities and techniques focus on the question: 'What does the solution need to accomplish before we consider a requirement implemented? User testing and validation of rapid prototypes (even before code) further develop the requirements with specific conditions of satisfaction that the solution must meet.

This includes the acceptance criteria that define the boundaries of a work item and are used to confirm when it’s completed.

PM²-Agile recommends two testing practices: Test Management and Concurrent Testing.

Test Management is an important concern for any sizeable software engineering effort. The Test Management practice provides a good starting point for practitioners relatively new to this area, but it also offers a structured reference model for the more seasoned ones. The tools used in planning, designing, implementing, executing, evaluating, and managing the test tasks or work products are an important aspect and need to align with the current organisation's practices.

The **Concurrent Testing** practice adopts testing concurrent with development throughout each iteration. This prevents teams from compressing testing into a separate activity at the end of iterations. Concurrent testing reinforces the concept of feature teams working in parallel. Concurrent testing requires a high degree of integration and high-bandwidth communication between developers and testers.

When addressing delivered increments testing, two points should be examined.

- Consider testing-related activities in parallel with requirements gathering and refinement so that the Agile Project Core Team (A-PCT) and the stakeholders can agree on specific conditions of satisfaction for each requirement. Ensuring acceptance criteria are defined for each Feature and Story allows the team to check if what is delivered meets information system users’ expectations.
- Run tests as frequently as possible. Ideally, all tests should be run against each build deployed to the mainline. If this is impractical, regression tests should be run for existing functionality while focusing the test cycle on work items completed in the new build. Even test scripts that are expected to fail provide valuable feedback. However, once a test script passes, it should not fail against subsequent builds of the solution.

## 4.14 Deployment & Transition

Regardless of the approach applied, there is always a need to ensure that the information system under development can effectively be deployed in the organisation's technical infrastructure. The Operational Model describes all these technical and infrastructure related aspects of deploying the information system.

There are several considerations to evaluate together with the technical aspect of deploying an information system to enable a smooth business transition. Activities, such as user testing, frequent training, and communication sessions also lead to an increased level of engagement. Ensuring that users are also continually informed and aligned with the project delivery schedule allow the optimal delivery of the solution.

Deployment and transition activities are also crucial to ensure that dependencies with other projects and/or information systems are taken into consideration. For further information on this topic, refer to PM² guide on managing Transitions and Business Implementation.

# 5 Ceremonies

When PM²-Agile is used in a project, a group of elements must be considered to increase the likelihood of its successful deployment. One of those elements is a set of critical ceremonies that help the Agile Project Core Team (A-PCT) achieving alignment in the following dimensions:

- Coordinate and Plan the work.
- How to execute the work.
- Inspect and adjust the work and the supporting processes.

These dimensions are materialized by the CIR rhythm[13]

The CIR rhythm (**C**oordinate, **I**mplement, **R**eview) is supported by a combination of ceremonies and engineering practices that were designed to ensure the Agile approach is structured and consistent. All the three aforementioned aspects play a crucial role.

The Coordination aspect (Coordinate) focuses on the ceremonies that guarantee alignment between the Agile Project Core Team (A-PCT) members on aspects like planning, what to build, priorities, etc. The PM²-Agile ceremonies that support this effort are:

- Iteration Planning
- Daily Stand-up
- Release Planning

The Implementation aspect (Implement) focuses on building the right solution and relies on several different engineering and co-creation techniques to achieve its goal (rather than on specific ceremonies). Because each Agile Project Core Team (A-PCT) has its own set of values, working agreements and particular skills, PM²-Agile does not explicitly suggest any specific tools or ceremony to help the team defining its way of working. Instead, it suggests the use of several tools and techniques, some of them detailed in the PM²-Agile Tools and Techniques guide.

The Reviewing aspect (Review) is focused on the activities and ceremonies used to assess the *usefulness* of the solution being built and the adequacy of the process used by the Agile Project Core Team (A-PCT).

To help achieving these goals, PM²-Agile recommends two key ceremonies:

- Iteration Review
- Iteration Retrospective

PM²-Agile acknowledges how important these events and activities are to enable the CIR rhythm. Nonetheless, the events and ceremonies described in this section are not a silver bullet and will not solve existing problems just because they have been implemented. It's the right set up of ceremonies and artefacts, with properly defined roles and responsibilities, supported by the correct tools & techniques and guided by the correct Mindsets that will increase the likelihood of a successful agile deployment.

The five ceremonies mentioned above are described in the following sections.

## 5.1 Iteration Planning

One of the main pillars of the PM²-Agile model is the Iterative Development. Iterative Development is a critical aspect in the complete lifecycle of an Agile project, enabled by the concept of Iteration that guides the development of the solution. By contributing with a validated solution increment, each iteration allows the solution to grow in an incremental way.

Because Iterations encompass a short period of time, a well-defined strategy and a plan on what will be developed are essential to ensure that the Agile Project Core Team (A-PCT) remains focused on meaningful delivery, iteration after iteration. To tackle this need, PM²-Agile recommends the setup of a specific ceremony: the Iteration Planning.

Iteration Planning aims to create a plan and a strategy that accounts for the team's availability and capacity to reach the agreed-upon goal.

How often the Iteration planning should occur and how long shall it take are also two important factors that must be carefully considered. One doesn't want to perform an iteration planning more often than it's necessary nor doesn't want to spend more time than is actually needed. As such, the way the Iteration

[13] For more information on the CIR rhythm, please refer to section 3.3.1-Lifecycle

Planning is structured and organized can make the difference between a successful ceremony and an unproductive event. Therefore, PM²-Agile recommends a specific structure with a set of guidelines that assist the team to successfully deliver such a ceremony.

### 5.1.1 Frequency and Duration

Designed to elaborate a strategy on reaching a specific goal in the short period of an iteration, the best moment to conduct the Iteration Planning is at the beginning of the iteration. PM²-Agile recommends the Iteration Planning as the first activity to be conducted by the Agile Project Core Team (A-PCT) within the iteration.

Another important aspect is the duration of the ceremony. The goal is to find the right balance between the invested time and the clarity/objectivity of the plan to be defined. However, since several factors such as the team and iteration sizes apply, universal rules to determine this balance cannot be applied. Therefore, PM²-Agile recommendation is based on the practical experience of thousands of teams that use both PM²-Agile and other methodologies and frameworks.

For a typical Agile Project Core Team (A-PCT) with five to nine Agile Team Members (ATeM), PM²-Agile suggests planning two hours for each iteration week. For example, planning four hours for eight Agile Team Members (ATeM) team, working in two weeks iteration is considered a good practice.

### 5.1.2 The Structure of an Iteration Planning

The Iteration Planning is there to help the Agile Project Core Team (A-PCT) plan the implementation of the Work Items List (WIL) iteratively. Because it reflects the work to implement a solution, the WIL must always be prioritized to allow the Product Owner (PrOw) proposing an Iteration Goal that is relevant and aligned.

When proposing the Iteration Goal, the Product Owner (PrOw) is communicating her expectations. With this information and considering capacity and capability, the Agile Project Core Team A-PCT) determines what they need to do, how they plan to do it and how long they expect it to take to validate if the proposed goal is achievable or not.

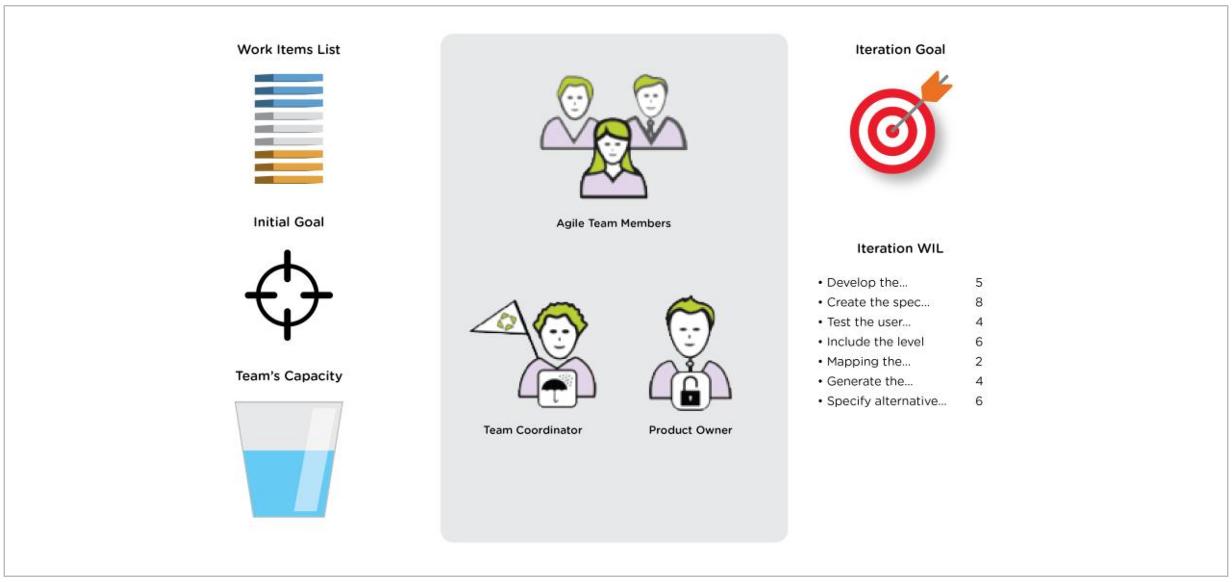


**Fig. 5.1** The Organics of an Iteration Planning

By the end of the iteration planning, the Agile Project Core Team (A-PCT) should have a clear and well-defined Iteration Goal and a generated iteration Work Items List, which are vital elements in the Iteration Plan.

After clarifying iteration planning organics, it's essential to understand proper implementation steps.

To help getting the best from the Iteration Planning, PM²-Agile proposes a sequence of steps that involves the entire Agile Project Core Team (A-PCT). These steps include the need for the team to determine its capacity, define the work to be done, have a temporary agreement on the iteration goal proposed by the Product Owner (PrOw) and have a vote of confidence.

#### *5.1.2.1 Determine Team's Capacity*

A crucial element in the Iteration Planning is understanding if the Agile Project Core Team (A-PCT) will be able to achieve the iteration goal. To verify this, each Agile Team Member (ATeM) building the solution must know how much time (in hours, preferably) will be able to dedicate to the next iteration. This exercise should also consider team members' holidays, public holidays and other events.

For instance, let's assume that each working day has seven hours of available work. Let's also say that an iteration of ten calendar days has only nine working days (one is a public holiday). To determine the Agile Team Member's capacity, one needs to compute the following data:

Capacity: 9 days x 7 hours = 63 hours

Therefore, the Agile Team Member (ATeM) will have sixty-three hours available to help the team reach the iteration goal.

Other aspects like unplanned events, bugs, corporate meetings, etc may also impact overall availability and should be considered as well. These aspects are described in the Guidelines section below.

#### *5.1.2.2 Agree on the Iteration Goal*

Defining the iteration goal is the most important aspect of an Iteration Planning, as this will keep the Agile Project Core Team (A-PCT) focused on what is important.

Defining the Iteration Goal is the Product Owner's (PrOw) responsibility and It must start with a prioritized Work Items List (WIL). The Product Owner (PrOw) evaluates the existing items and formulates an iteration goal that is meaningful and adds value to the solution.

Once the goal is defined, the Product Owner (PrOw) relays it to the rest of the Agile Project Core Team (A-PCT) to focus on the most relevant items.

Note that an initial agreement does not imply that the iteration goal will be immediately approved as the Agile Team Members (ATeM) that develop the solution need to confirm their availability and capacity.

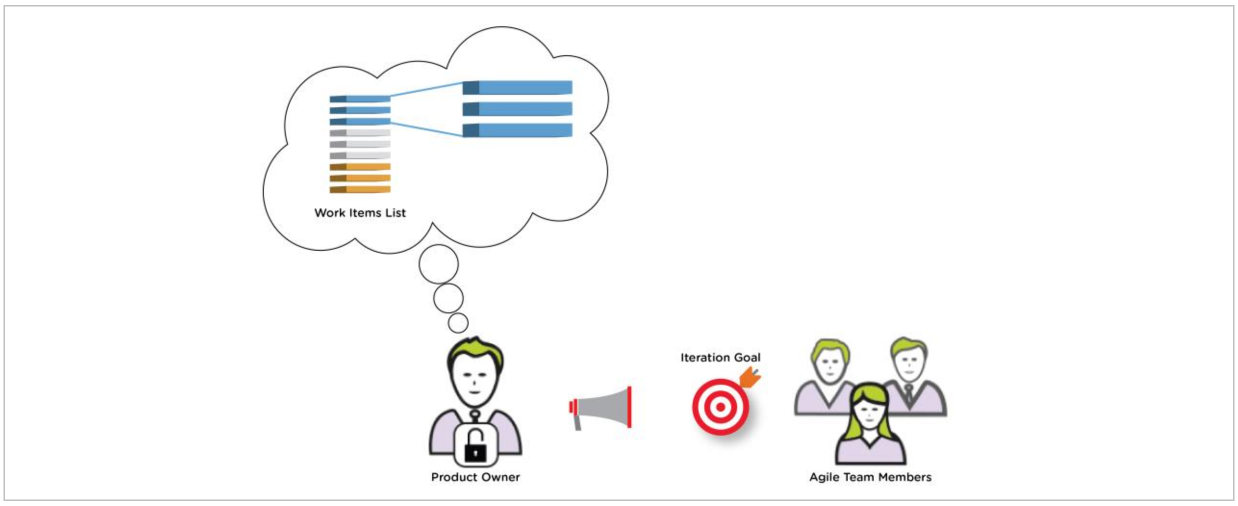


**Fig. 5.2** Agree on the iteration goal

#### *5.1.2.3 Define the Work to be Done*

With the team's availability and the iteration goal defined, the Agile Project Core Team (A-PCT) can determine the work that needs to be done. The team should focus on the items in the Work Items List (WIL) that contribute to the iteration goal. For each item, they determine the tasks they need to execute and verify they have the availability and capacity to complete them.

They will perform this routine for each relevant item from the WIL until they exhaust their capacity.

To Define the Work to be Done, PM²-Agile recommends the following steps:

1. The Agile Project Core Team (A-PCT) picks up the first item in the WIL related to the iteration goal.
2. As a team, they identify the tasks they believe are required to build and deliver that item.
3. Each task is chosen by an Agile Team member (ATeM) for implementation.
4. The Agile team member (ATeM) estimates the task (should be no bigger than 12 hours).
5. Each Agile team member (ATeM) verifies availability after taking the task.

6. If every Agile team member (ATeM) still has availability, the team picks up the next Work Item from the Work Items List (WIL).
7. If an Agile team member (ATeM)  verifies the task cannot be delivered, other team members perform a check. If no other team member can perform the task, the team should stop progress.
8. Once the team reaches their full capacity, it's time to to evaluate if the iteration goal can be reached or it needs to be adjusted. This is the goal of the next step: the Commitment Agreement.

#### *5.1.2.4 Commitment Agreement*

The Commitment Agreement formalizes agreement between the Product Owner (PrOw) and the rest of the Agile Team Members (ATeM) per the Iteration Goal. This agreement is critical, since it aligns everyone's expectations transparently in terms of team delivery.

There are three possible scenarios in initiating the Commitment Agreement. The first (and simplest) is when the team confirms (based on the information they collected during the previous steps) they can reach the Iteration Goal.

In the second scenario, the Agile Team Members (ATeM) find they cannot achieve the Iteration Goal. A discussion with the Product Owner (PrOw) allows the team to propose changes to the iteration goal to make it less ambitious and thus, achievable. The Product Owner (PrOw) will evaluate the feedback given by the team and will try to adjust accordingly.

In the third scenario, the Agile Team Members (ATeM) find they can deliver beyond the Iteration Goal. A discussion with the Product Owner (PrOw) allows the team to confirm that the Work Items delivered beyond the Iteration Goal are indeed the most relevant ones.

#### *5.1.2.5 Vote of Confidence*

Many aspects, opinions and adjustments are shared and discussed during an Iteration Planning ceremony, which can considerably change the initial strategy and direction. As such, it's very important that all Agile Team Members (ATeM) take a couple of minutes to think about the proposed strategy and express how comfortable they are with it.

To facilitate the vote of Confidence, each Agile Team Member (ATeM) and the Product Owner (PrOw) will use a finger count to choose a value from 1 to 5 reflecting confidence in the proposed plan. A "1" means "No confidence at all" while a "5" means "I'm 100% that we will complete it successfully".

The steps are the following:

1. Agile Team Members (ATeM) and Product Owner (PrOw) take a minute to assess their confidence.
2. On the count of three, everyone will raise the hand at the same time.
3. Initially, the facilitator asks the "5's" to lower their hands,  followed by the "4's" and finally the "3's".
4. Team members who rate "2's" or "1's" should explain their concerns and doubts.
5. The entire Agile Project Core Team (A-PCT) will then perform the required adjustments (if any) to address those concerns.

Once all the concerns are discussed and there are no more "1's" or "2's", the Vote of Confidence ends and with it, the Iteration Planning.

### 5.1.3 Guidelines and Participants

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates the execution of the iteration planning, ensuring the team gets the best out of it. Assists the team keeping the time slot and discovers any anti-patterns that may jeopardize the success of the meeting. |
| Agile Team Member (ATeM) | Responsible for the execution of the Iteration Planning, including the setup and building of the Iteration Work Items List (WIL). |
| Product Owner (PrOw) | Supports the execution of the Iteration Planning by proposing an Iteration Goal and providing all the necessary clarifications requested by the Agile Team Members (ATeM) when building the Iteration Work |

| | |
|---|---|
| | Items List (WIL). Accountable for the iteration goal agreed (output) with the Agile Project Core Team. |
| Architecture Owner (ArOw) | Besides being also an Agile Team Member (ATeM), the Architecture Owner (ArOw) supports the Product Owner (PrOw) with the Iteration goal definition when enablement work is involved by clarifying potential dependencies or mitigating Architectural risks. |
| Project Manager (PM) | Is informed about the Iteration Goal and the plan defined by the Agile Project Core Team (A-PCT) to achieve it. |

#### *5.1.3.1 Guidelines*

- **Set time aside for unplanned events** – Because Bugs in Production, support required or other unplanned events cannot be foreseen, the Agile Project Core Team (A-PCT) should set some time aside to deal with those. For instance, 10% dedicated to unplanned events means that after calculating each team member's availability, an additional 10% is taken, leaving the Agile Team Members (ATeM) with less time available to deal with the planned items.
- **Assess Agile Team Members's (ATeM) capacity in advance** – If every team member previously determines individual capacity, the team can save some time during the Iteration Planning. When doing this, the Team Coordinator (TeCo) should make clear the operating rules on this procedure, especially when additional time for unplanned events is required.
- **Have the Work Items List (WIL) prepared** – A properly prioritised Work Items List (WIL) can make the difference between an intense but effective Iteration Planning and a deceptive and demoralizing ceremony. Ensure the Product Owner (PrOw), supported by the Team Coordinator (TeCo) and the Architecture Owner (ArOw) performs this task.
- **Have the Acceptance Criteria defined** – The Work items the team has previously refined and estimated must have Acceptance Criteria. This helps the team defining their implementation strategy.
- **Have an extra two or three work items ready –** After the team reaches its full capacity and finalises the confidence vote, decompose another two or three items. They will not be included in the iteration but will be ready to be tackled in case the team has extra time or a work item is dropped during the iteration.
- **"Difficult-to-find" tasks –** When decomposing an item from the Work Items List (WIL), there will be a moment where the team will struggle to find more tasks. When this happens, they should move to the next work item.
- **Using Yesterday's weather** – This concept refers to using the team's past velocity to determine their current capacity. Although PM²-Agile doesn't recommend this practice, using past velocity can be useful for the Product Owner (PrOw) to evaluate how realistic the Iteration Goal is by comparing that value with the amount of points given by the work items foreseen in the Iteration.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Iteration Planning | I | I | **S** | **S/A** | **S** | **R** |

## 5.2 Daily Stand-up

The daily routine of a project following agile values and principles relies on communication as a fundamental aspect. The supporting principles of PM²-Agile make it quite clear that without proper communication, any project will be deemed to fail. To avoid such a scenario, PM²-Agile recommends creating a mechanism to keep the Agile Project Core Team (A-PCT) aligned towards its iteration goal and identifying potential obstacles. In PM²-Agile, this mechanism is known as the Daily Stand-up. During this ceremony, the Agile Team Members (ATeM) check the heartbeat of the project and getting individual commitments from each other until the next checkpoint (usually, twenty-four hours).

The Daily Stand-up is neither a status update meeting nor a problem-solving meeting. Issues raised are registered and discussed by the relevant team members immediately after the meeting.

### 5.2.1 Frequency and Duration

The first question to address is: How often should this check-up occur? Daily? Once a week? Three times a week? In theory, the answer is "it depends" since the frequency is determined by team's dynamics and the work being performed. However, with embedded Agile values and principles, those dynamics urge a more frequent check-in. Therefore, PM²-Agile suggests a daily Stand-up ceremony.

The duration of the ceremony relates to the number of Agile Team Members (ATeM) participating in the meeting. As a rule of thumb, PM²-Agile suggests sixty to ninety seconds for each Agile Team Member (ATeM) to share relevant information and guarantee alignment with the team. With a team of ten Agile Team Members (ATeM), the maximum duration should be about fifteen minutes.

### 5.2.2 The Structure of a Daily Stand-up

Though the Daily stand-up is a reasonably short and simple ceremony, it is as vital as other meetings in PM²-Agile.

The Daily Stand-up aims to ensure the Agile Project Core Team (A-PCT) is aligned on what is being done and what needs to be done to reach the iteration goal. Because each Agile Team Member (ATeM) must provide updates clearly and objectively, PM²-Agile suggests that each Agile Team Member (ATeM) considers the following questions:

- What was I working on since the last checkpoint towards the iteration goal?
- What will I be working on until the next checkpoint towards the iteration goal?
- What blockers do I have that may prevent us to achieve the iteration goal?
- How confident am I that we will accomplish the iteration goal?

The Daily Stand-up starts spontaneously when an Agile Team Member (ATeM) takes the initiative and starts sharing. As she finishes, the next team member should follow. As information is shared, topics may emerge that require a more detailed analysis or evaluation, significantly if the iteration goal is jeopardised. In this case, the Team Coordinator (TeCo) notes which Agile Team Members (ATeM) should participate and registers them for a "meet after". After everyone has shared their information, the Team Coordinator (TeCo) reminds the participants of their "meet after" so they can make the necessary arrangements. It's the Agile Team Members' (ATeM) sole responsibility to organise the "meet after", not the Team Coordinator (TeCo)

### 5.2.3 Guidelines and participants

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates the execution of the Daily Stand-up. Ensures that each Agile Team Member (ATeM) has the chance to speak and respects the guidelines. Supports the consolidation of the "meet-after" identified during the meeting. |
| Agile Team Member (ATeM) | Responsible for the execution of the Daily Stand-up, provides information about the work being done and how it's aligned with the rest of the team and the Iteration Goal. |
| Product Owner (PrOw) | Informed about the outcomes of the meeting only if necessary. |
| Architecture Owner (ArOw) | In this particular case, acts like a regular Agile Team Member (ATeM). |
| Project Manager (PM) | Is informed about the outcomes of the Daily Stand-up |

#### *5.2.3.1 Guidelines*

- **Same time, same place** – The Daily Stand-up is part of the daily routine of the Agile Project Core Team (A-PCT). Holding it in the same place and time helps consolidating this routine and allows all the participants to organize their agendas in a more structured and optimized manner.
- **Prepare communication** – Though It's a short meeting, an Agile Team Member (ATeM) should dedicate a few minutes to prepare communication properly.
- **Keep everyone focused** – When a meeting is not prepared, it's very easy to lose focus. When this happens, the Team Coordinator (TeCo) should guide the Agile Team Member (ATeM) back on-track.

- **Use a Kanban board[14] to illustrate communication** – Based on the principle that an image is worth a thousand words, every Agile Team Member (ATeM) should, whenever possible, materialize her communication using elements from the Kanban board (either physical or digital).
- **Use the Parking lot to manage "pendencies"** – To make visible the "pendencies" identified during the Daily Stand-up, the Team Coordinator (TeCo) can use a pre-defined space in the Kanban board (called the "Parking Lot"). Note the name of the topic and one or two names involved on a post-it to create visibility. By the end of the Daily Stand-up, the team will see future activities and who's involved.
- **Use common sense** – Although the Daily Stand-up is a specific meeting, normal common-sense rules should apply. Do not interrupt, listen carefully, be honest and transparent are some of the basic rules.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Daily Stand-up | N/A | I | **S** | I | **R** | **R/A** |

## 5.3 Release Planning

People live today at a completely different pace they used to twenty, fifteen or even ten years ago. They process a lot of information and make thousands of decisions every day. The word VUCA[15], used more than ever to describe the world people live in, is just one of the several pieces of evidence that things have changed.

Currently, technological, economic, social and political aspects determine that what was true yesterday, may no longer be true today, either because something changed, it was not clearly understood or was too complex to maintain. This is the new reality and it's impossible to avoid it. Moreover, the organisations that manage to adapt are the ones thriving.

When building a solution, a team should be able to refer to an organisational strategy (marketing focused, risk oriented, opportunity enablement) that clearly defines what, when and to whom will be a solution delivered. Because the strategy drives the team throughout the entire project, it is paramount to guarantee that it reflects what the business (and the organisation) needs. Still, this is just the beginning. With all the dynamism that characterises the VUCA world, the business must have the tools that will allow them to share their strategy with the Agile Project Core Team (A-PCT) regularly. This is where PM²-Agile suggests the use of Release Planning.

Release Planning is designed to materialise the Learning step of the PM²-Agile Lean Start-up model. By looking at the collected data, it's possible to learn and generate new ideas that will drive the building of the solution. By practicing two critical steps of the iterative PM²-Agile Lean UX process: Evaluate and Learn and Incept assumptions and research, the Agile Project Core Team (A-PCT) and all the other relevant business stakeholders focus on essential assumptions that need to be validated.

Release planning's primary goal is to guarantee that the business communicates to the Agile Project Core Team (A-PCT) ongoing changes that affect the solution strategy. The team determines how "big and complex" those changes are so the business can decide what to build and when to release.

Allowing a release strategy to become obsolete can cause total or partial unsuitability of the solution with negative consequences. Therefore, the question to be asked about Release Planning is not "shall it be used" but instead "how often shall it be done?"

### 5.3.1 Frequency and Duration

Changes in a solution's release strategy have different reasons or justifications. Still, they always arrive in the context of the following perspectives:

- **Top-down** – A change in the organisation's strategy as a response to a change in its context.
- **Bottom-up** – Based on the feedback of the Product Owner (PrOw) or an Agile Team Member (ATeM) as the solution continues to evolve.

[14] For more information on Kanban board, please check the technique "Kanban Method"

[15] **V**olatility, **U**ncertainty, **C**omplexity, **A**mbiguity – Was first introduced in 1991 by the U.S. Army as a result of the extreme conditions in Afghanistan and Iraq. Today's business environment has changed in a very similar way.

Regardless of perspective, there are three aspects to consider in Release Planning to guarantee the desired value.

- **Manage the Work Items List (WIL)** –Product Owner (PrOw) and other business stakeholders decide what new elements may need to be added, updated or removed from the Work Items List (WIL).
- **Work Items List (WILL) refinement** – The Product Owner (PrOw) and the rest of the Agile Project Core Team (A-PCT) review the information coming from the Manage the WIL aspect and provide, when needed, new estimates. They also detail and validate the next Work Items the team will work on, review estimates, etc. Additional stakeholders like the Business Manager (BM) and the Users Representatives from the Business Implementation Group (BIG) may also join as needed.
- **Work Items List (WIL) Prioritisation** - Product Owner (PrOw) and Stakeholders review the Work Items List (WIL) and redefine the priorities.

PM²-Agile recommends this sequence of steps to be performed at least once per iteration (considering a 2-week iteration). However, the longer an iteration is or the more volatile the solution context is, the more frequent the release planning should be.

One may ask if these steps should be performed as a single ceremony or as separated ones. Experience shows that organising these steps as individual ceremonies yields better results than executing a single one.

PM²-Agile does not have a specific recommendation for the duration because for each step, different tools and techniques can be used. For instance, user story workshops in the first step are quite common and can last several hours compared to an approach where only one person identifies and presents the stories to the rest of the stakeholders. Design blocks[16] yield great results for both the first and second steps. The goal is to find a balance between the time invested and each step's objective.

### 5.3.2 Structure of the Release Planning

The main goal of Release Planning is to adjust the release strategy according to the changes in the organisational context (top-down) and/or in the context of the daily work and feedback from the Product Owner (PrOw) (bottom-up).

Effective Release Planning must consider the following activities:

- Manage the Work Items List (WIL)
- Work Items List (WIL) refinement
- Work Items List (WIL) Prioritisation

Each step has a specific purpose, involves different actors and generates different outputs. A prioritised Work Items List (WIL) reflecting the updated reality and a shared understanding of the vision will allow the Release Plan to be updated as a result.

The following sections describe in more detail each of these steps.

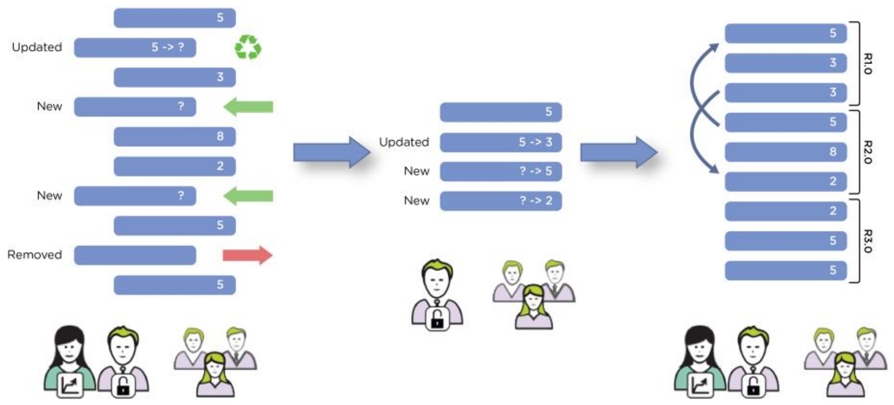


**Fig. 5.3** The three key steps in Release Planning

[16] For more information on Design Blocks, please check the "Design Blocks" tool/technique.

#### *5.3.2.1 Manage the Work Items List (WIL)*

Both the Product Owner (PrOw) and the Business Manager (BM) should align and share the same vision in regards to changes that will likely have an impact in the solution's Work Items List (WIL). The Lean UX process starts a new cycle by incepting new and challenging current assumptions. This assures the Agile Project Core Team (A-PCT) will invest their time in the most valuable things.

Alignment between the Product Owner (PrOw) and the Business Manager (BM) can be achieved in this Manage the Work Items List activity. Other stakeholders may also need to be involved in this discussion without focusing on the Work Items List (WIL) but rather on features that will be delivered to fulfill the identified needs. To support this, the Product Owner (PrOw) can create a separate ceremony.

As soon as the Product Owner (PrOw) and the Business Manager (BM) are aligned, they will work together with the Business Implementation Group (BIG) to define how the Work Items List (WIL) will materialise the agreed-upon changes. New items can be added, others removed or even updated. Whatever the case is, their goal and acceptance criteria must be clear.

As a tool to support this activity, Design Blocks bring together the Agile Project Core Team (A-PCT) and all the relevant business stakeholders to discuss and challenge assumptions, formulate hypotheses and define different ways to test them. PM²-Agile recommends using Design Blocks, supported by Collaborative design, including design studios and Design Systems to encourage alignment.

PM²-Agile recommends that the Team Coordinator (TeCo) and an Agile Team Member (ATeM) with UX Design skills facilitate and participate in this ceremony when using tools like the ones mentioned.

The final output should be a reviewed and prioritised Work Items List (WIL), according to the Product Owner's (PrOw), Business Manager's (BM) and Business Implementation Group (BIG) expectations. It will be the next activity'starting point: the Work Items List (WIL) refinement.

#### *5.3.2.2 Work Items List (WIL) refinement*

After preparing the Work Items List (WIL) with the Business Manager (BM), the Business Implementation Group (BIG) and other stakeholders, the Product Owner (PrOw) describes all the changes to the rest of the Agile Team Members (ATeM). In return, the team will provide valuable information regarding the changes' feasibility and size (estimates).

The Product Owner (PrOw) and the Agile Team Members (ATeM) will also review the Work Items that are expected to be part of the next iteration and make sure they all come to a common understanding and agreement. Design Blocks play, once again, a vital role in this activity.

Briefly, the two key activities in a Work Items List (WIL) refinement session are:

- Discuss the most relevant items from the Work Items List (WIL).
- Analysing and reviewing estimates.

One must determine what are the most relevant items from the Work Items List (WIL). First, the items that have the highest priority and will be included in the next iteration planning and secondly, the items that were reviewed in the previous Manage the Work Items List (WIL) session.

Once identified the Work Items, the Product Owner (PrOw) should discuss them with the rest of the Agile Project Core Team (A-PCT).

More level of detail should be added to the Work Items that are expected to be included in the next Iteration Planning. This activity is a very important enabler for a successful Iteration Planning Meeting, since the team will need to determine how to build stuff.

On the other hand, the new or reviewed items that will be implemented a few iterations ahead require less details. Nevertheless, it needs to be clear enough to allow the Agile Team Members (ATeM) to understand the concept, dependencies and logical links with the other items in the Work Items List (WIL).

This implementation order is essential because one doesn't want to invest time on non-priority items far from being implemented.

The Work Items List (WIL) refinement activity materialises the Incept assumptions and research step of the PM²-Agile Lean UX cycle towards an optimal solution. These discussions will help determining how to test the hypotheses that were formulated and need validation. The use of Design Blocks is a great way to support these activities.

While the Work Items List refinement inputs are mostly coming from the Product Owner (PrOw), the rest of the Agile Project Core Team (A-PCT) can also identify items to be included in the Work Items List (WIL). This is the case of technical items (enablers), where the Architecture Owner (ArOw) explains to the Product Owner (PrOw) how important they are to support the implementation of the overall solution.

After discussing and understanding all the work items, the Agile Project Core Team (A-PCT) should review and provide relative estimates. This allows the Product Owner (PrOw), Business Manager and other stakeholders to engage in a well-educated prioritisation exercise, leading the next activity: Work Items List prioritisation.

#### 5.3.2.3 *Work Items List (WIL) prioritisation*

As the final step in the Release Planning cycle, the Product Owner (PrOw) and all other stakeholders adjust the release strategy for the final users' solution by prioritising the Work Items List (WIL).  This simple activity is the main outcome of Release Planning.

Based on the estimates provided by the Agile Team Members (ATeM) during the Work Items List (WIL) refinement activity, the Product Owner, aligned with the Business Manager (BM) and other stakeholders, prioritises the several items and reviews the content of the several planned releases, updating, when applicable, the current Release Plan.

During this activity, the Product Owner (PrOw) and the other stakeholders examine the items identified by the remaining Agile Team Members (ATeM), paying attention to enablers, which may compromise the integrity and usability of the solution, if not implemented when planned. Because neither the Product Owner (PrOw) nor the remaining stakeholders are familiarised with most of the enablement work items, PM²-Agile recommends the presence of the Architecture Owner (ArOw) to help with the prioritisation.

By the end of this activity, a revised release plan is generated and shared with the entire Project Core Team (PCT). Depending on the impact, the modified release plan may need to be reviewed by the Project Steering Committee (PSC) to agree on the changes.

### 5.3.3 Guidelines and Participants

Release Planning identifies three critical activities with different purposes, both in what they aim to achieve and with respect to various participants' roles. As such, and to make this information clearer, the guidelines and participants are described in the context of each activity.

#### 5.3.3.1 *Manage the Work Items List (WIL)*

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates the execution of this activity, especially if it contains tools like Design Blocks, User-story workshops or User-story mapping. The Team Coordinator (TeCo) guarantees the Work Items List (WIL) reflects the agreed changes by the end of the activity. |
| Business Manager (BM) | Supports the execution of this activity by explaining how the changes/adjustments to the solution or organisation context may impact the Work Items List (WIL). |
| Agile Team Member (ATeM) | Informed of the outputs of this activity. The Agile Team Member (ATeM) can also support this activity by facilitating and conducting Design Blocks sessions if she possesses relevant experience. |
| Product Owner (PrOw) | Responsible for the execution of this activity. Creates and updates all the work items needed to complete the Business Manager's (BM) changes and is accountable for the ceremony's outputs. |
| Architecture Owner (ArOw) | Informed of the outputs of this activity. |
| Business Implementation Group (BIG) | The User Representatives that are part of the Business Implementation Group (BIG) are consulted by the Product Owner (PrOw) regarding the changes foreseen to the Work Items List (WIL) |
| Project Manager (PM) | Informed of the outputs of this activity. |

***Guidelines***

- **Keep a close eye on the Features** – Features are the answer to the Needs to be fulfilled; they need to be organised and prioritised as part of the solution's release strategy. This will ensure that the Manage the Work Items List activity will be aligned with the global strategy.
- **Make it Visual** – Visualisation is critical when discussing ideas, assumptions, hypotheses to test, etc. Design Blocks provide the environment, the playground and additional tools to ensure that all the stakeholders come to a common understanding and agreement on what, how and why things need to be done.
- **Define the tools to use in advance** – The definition and acquaintance with the several tools that can be used to explore and evolve the Work Items List (WIL), Conducting a workshop where people are not familiarised with the concepts can be very frustrating.
  The Team Coordinator (TeCo) can play a key role in explaining the tools' dynamics. Agile Team Members (ATeM) with specific skills may also provide invaluable assistance.
- **Prepare the session in advance** – These ceremonies can be quite intense, with a lot to cover. Previous research on specific topics is highly recommended to ensure the participants focus during the session.
- **Know the Work Items List (WIL)** – Getting familiarised with the Work Items List (WIL) is part of the daily work of the Product Owner (PrOw). Knowing the Work Items List (WIL) helps analysing if new items must be added, existing items must be removed or potential dependencies between work items.

| **RAM** (RASCI) | **BIG** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|---|
| Manage Work Items List (WIL) | C | **S** | I | **S** | **A/R** | I | I/**S** |

#### *5.3.3.2 Work Items List (WIL) refinement*

| **Key Participants** | **Description** |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates this activity, ensuring that the team understands and agrees on each item's acceptance. Assists the team conducting estimation exercises, making sure they adhere to the rules. |
| Business Manager (BM) | Informed by the Product Owner (PrOw) of the outputs of this activity. |
| Agile Team Member (ATeM) | Supports this meeting's execution by contributing with relevant data regarding estimates and with new items when necessary. Accountable for the output of the activity. She also supports this activity by facilitating and conducting Design Blocks sessions. |
| Product Owner (PrOw) | Responsible for the execution of this activity by explaining the relevant items. |
| Architecture Owner (ArOw) | Contributes to the activity as an Agile Team Member (ATeM). |
| Project Manager (PM) | Informed of the outputs of this activity. |

***Guidelines***

- **Share in advance with everyone all the items to be discussed** – The Product Owner (PrOw) should share the items she expects to discuss with the rest of the Agile Project Core Team (A-PCT), allowing the team to be already prepared and focused.
- **Explain and discuss first, estimate after** – Before proceeding to the estimation exercise, make sure that all the items have been adequately discussed with the Agile Team Members (ATeM). This promotes understanding and details possible links and dependencies between the items, which add value while providing relative estimates.
- **Make it Visual** – Visualisation is critical when discussing ideas, assumptions, hypotheses to test, etc. Design Blocks provide the environment, the playground and additional tools to ensure that all the stakeholders come to a common understanding and agreement on what, how and why things need to be done.

- **Be precise on short term, concise on medium and long term** – Invest more time detailing short term work items and be more concise but clear for medium and longer-term items.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Work Items List (WIL) refinement | I | I | **S** | **R** | I | **S/A** |

#### *5.3.3.3 Work Items List (WIL) prioritisation*

| **Key Participants** | **Description** |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates the execution of this activity when requested by the Product Owner (PrOw). |
| Business Manager (BM) | Supports the Product Owner (PrOw) determining the priorities of the Work Items List (WIL) |
| Agile Team Member (ATeM) | Consulted by the Product Owner (PrOw) to validate the new release plan. |
| Product Owner (PrOw) | Responsible for the execution of this activity. Together with the Business Manager (BM), determines the priority of the Work Items List. Accountable for the output of this ceremony. |
| Architecture Owner (ArOw) | Consulted during the activity to make sure all the questions and doubts regarding the Work Items List's enablement items are clarified. |
| Business Implementation Group (BIG) | Consulted as the User Representatives to provide inputs on the prioritisation of the Work Items List (WIL). |
| Project Manager (PM) | Informed of the outputs of this activity. |

***Guidelines***

- **Take dependencies into account** – It may not be possible to implement the Work Items List (WIL) items independently. In this case, the recommended approach is to consult the Architecture Owner (ArOw) to understand better how those links impact the prioritisation.
- **Finalise with a clearly prioritised Work Items List (WIL)** – Reflect the release strategy in terms of solution and provide clear guidelines to the Agile Project Core Team (A-PCT) on what to focus next towards the implementation of the Release Plan.
- **Update the Release Plan** – After deciding on the prioritisation, information must be shared with the rest of the Agile Project Core Team (A-PCT) to update/validate the Release Plan while accounting for current velocity.

| **RAM** (RASCI) | **BIG** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|---|
| Work Items List (WIL) prioritisation | C | **S** | I | **S** | **R/A** | C | C |

## 5.4 Iteration Review

As described in the previous sections, each iteration kicks-off with an iteration goal defined by the Product Owner (PrOw). This triggers the rest of the Agile Project Core Team (A-PCT) to build a strategy they believe will help them reach the goal. Throughout the iteration, the team will implement this strategy by developing all the relevant items.

Unfortunately, not everything is black or white, and misunderstandings are common in creative activities like software development. As such, it is imperative to continually assess the work developed by the Agile Team Members (ATeM) to ensure alignment with the Product Owner's (PrOw) expectations. In this context, PM²-Agile recommends the use of the Iteration Review ceremony.

The Iteration Review's primary purpose is to assess if the iteration goal was achieved, which can be done with a practical demo of what the Agile Team Members (ATeM) built. This allows the Product Owner (PrOw) and other stakeholders to provide feedback and helps the Agile Team Members (ATeM) understand if they are going in the right direction.

Therefore, most of the time invested preparing the Iteration Review should focused on this demo.

The following sections will detail how a typical Iteration Review meeting must be structured and how often should it happen.

### 5.4.1 Frequency and Duration

The Iteration Review is strongly related to the Iteration Planning meeting described in the previous sections, connecting the planning and the reviewing activities of an iteration cycle.

At the end of the iteration, the Iteration Review meeting evaluates how successful the Agile Project Core Team (A-PCT) was in reaching the iteration goal. PM²-Agile recommends the Iteration Review as a closure ceremony meeting occurring on the last day of the iteration.

Several factors can influence the duration of the meeting. For instance, a longer iteration means, in theory, the team will develop and deliver more work, implying a more extended demo. Furthermore, when more work is delivered, the likelihood of finding or discussing more issues will probably increase and with it, the time required to discuss them.

So, although there is no established duration for the Iteration Review ceremony, the Agile Project Core Team (A-PCT) should strive for objectivity during the demo. Based on the field experience and assuming a 2-week iteration, PM²-Agile considers 90 to 120 minutes a good practice.

### 5.4.2 Structure of an Iteration Review

Following the agile principle “Working software is the primary measure of progress”, the core part of this ceremony is a demo conducted by the Agile Team Members (ATeM) where they expose the iteration’s achievements to the business stakeholders. However, to ensure a successful ceremony, preparation needs to be done, conclusions need to be taken and a set of actions to be planned as the outcome of the Iteration Review.

To ensure an effective meeting, the Iteration Review has three different moments that must be carefully prepared:

- **Align/Review Expectations** - A brief ten to fifteen minute presentation to align the expectations on the contents of the Demo. Should include a comparison between what was promised versus what was delivered.
- **Demo** - Following a specific set of guidelines, either the Product Owner (PrOw) or another Agile Team Member (ATeM) team member, demonstrates the progress achieved during the iteration.
- **Wrap-up + Work items List (WIL) Review** - After the demo, conclusions should be reached, followed by a discussion on impacts to the Work items list (WIL).

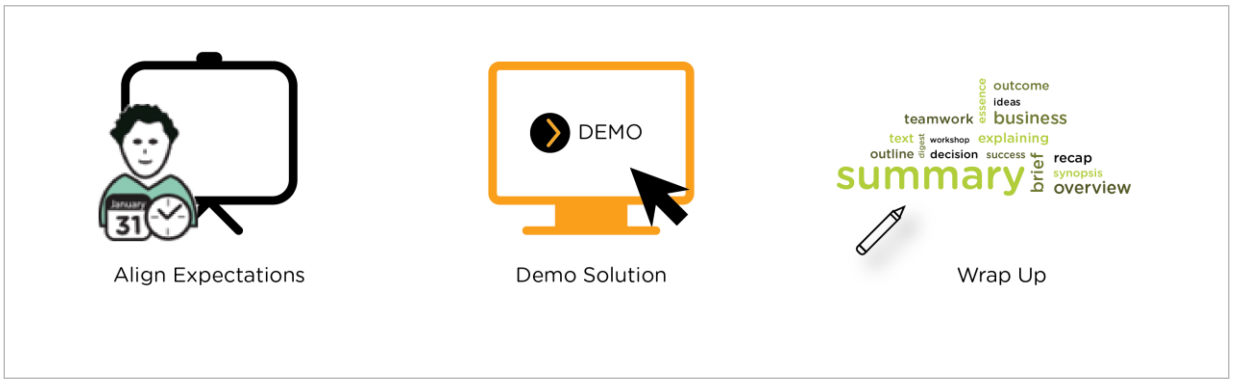


**Fig. 5.4** The structure of an Iteration Review

These moments are further explained in the following sections.

#### *5.4.2.1 Align Expectations*

The Agile Project Core Team (A-PCT) seeks to avoid misaligned expectations with the business stakeholders, especially during an Iteration Review. Misalignment leads to frustration leading to a lack of objectivity, which may affect the meeting’s outcome. Therefore, PM²-Agile recommends introducing one of the Agile Team Members (ATeM) to tackle the key points that will drive the demo. This is a 10 -15 minutes presentation that must address the following items:

- **Scope -** Describe the Iteration goal and the corresponding Work Items that the Agile Project Core Team (ATeM) committed at the iteration's start.
- **Achieved** - The work items the Agile Project Core Team (A-PCT) believes it has completed and will demonstrate.
- **Not Achieved** - The work items the Agile Project Core Team (A-PCT) could not deliver. This can be discussed unless related to risks and issues that are discussed after.
- **New/Dropped Items** - It's imperative to communicate this information to the business stakeholders, especially if new items were added to the work items list (WIL).
- **Difficulties/Impediments/Risks** - Quickly identify these items and describe how they affected the team and impacted Work Items.

With this quick introduction (which will likely trigger some questions), the Agile Project Core Team (A-PCT) ensures that everyone is now on the "the same page" on the Iteration Review's ceremony. This activity ensures everyone is ready for the demo.

#### *5.4.2.2 Demo the Solution*

The demo is, on its own merit, the most critical moment of this ceremony. This is when the Agile Project Core Team (A-PCT) exposes the business stakeholders and the Product Owner (PrOw) to the iteration's outcome.

In order to properly facilitate and conduct a demo, some decisions need to be taken and agreed in advance.

- **Single point demo or hands-on demo** - A single point demo means that the Product Owner (PrOw) or another Agile Team Member (ATeM) will conduct the demo, while a hands-on session will allow the business stakeholders to "feel" and use the solution being demonstrated. PM²-Agile does not suggest a specific option, since it will depend on the stakeholders' will and commitment. Having the stakeholders using and trying is optimal for demo delivery. However, a hands-on approach requires more effort to facilitate and conduct the demo, as everyone must be focused on what is necessary to validate during the entire session.
- **Who will be conducting the Demo** – The Product Owner is a good candidate to conduct the demo. She is empowered and closely involved with the business stakeholders, who may feel more comfortable if she is conducting the demo. However, it is also a good approach if an Agile Team Member (ATeM) conducts the Demo, as the Product Owner (PrOw) becomes available to assist the stakeholders during the session.
- **Decide which test scenarios to use** –The test cases created by the Agile Team Members (ATeM) during development should be used to evaluate the demo. However, the business stakeholders may want to create and user their own scenarios. If that's the case, the Agile Project Core Team (A-PCT) has to guarantee that those tests are aligned with the acceptance criteria of the stories, since that's what will guide the demo. Nevertheless and whatever the scenario is, this must be previously agreed so that the meeting facilitator can prepare the meeting properly.

With these elements properly defined and agreed, it should be clear for the Agile Project Core Team (A-PCT) who and how the demo will be facilitated.

Even with a well-defined facilitation plan, it's also very important to have a proper structure for the demo. Since several items are presented, each with its own acceptance criteria, its own test cases and with several different questions that may pop-up, the Agile Project Core Team (A-PCT) needs to guarantee that all these items are properly demonstrated, and the feedback is captured. To help achieving this, PM²-Agile foresees the following sequence of steps for each demonstrated item.

- **Identify and describe the item** – The person conducting the demo starts by introducing the item that will be demonstrated. This includes the description of the work item and the corresponding acceptance criteria. She can conclude with the indication of the several test cases that will be executed.
- **Demo the Item** – While demoing the item, each test scenario should be properly identified, including the acceptance criteria (very important) that the scenario is targeted to validate. This helps the stakeholders to clearly see the usefulness of the tests being made and will give them more confidence when they need to accept the item.
- **Ask for confirmation of the item** – If all the acceptance criteria are met, the Agile Team Members (ATeM) should ask for the confirmation that the item does exactly what was agreed to. In other words, to mark that item as DONE. If, on the other hand, some acceptance criteria are not met or there are new things to be made, they should be noted and discussed in the wrap-up step.

As mentioned before, these steps must be repeated for each item that needs to be demonstrated. As soon as all the items are done, the ceremony can proceed to the next step: Wrap-up and Work items List (WIL) Review.

#### *5.4.2.3 Wrap up and Work Items List (WIL) review*

During the solution demo, a lot of information is exchanged. As such, it's very important to leave enough time for consolidation, since these conclusions are used to keep a refined and prioritised Work Items List (WIL) and to prepare for the next Iteration Planning meeting.

After the demo is finished, an Agile Team Member (ATeM) or even the Team Coordinator (TeCo) must go through each demonstrated item and check the following:

- Was the Item Accepted or not? If not, it should be clear what was the reason (acceptance criteria not verified, bug was found, etc).
- Is it a new Item? Is it an Item that will be dropped from the Work Items List (WIL)?
- For the new items, clarification must be requested to the Product Owner (PrOw) and stakeholders. This will allow the Agile Team Members (ATeM) to provide high level estimates during the Work Items List (WIL) review that comes immediately after.

Once the Agile Project Core Team (A-PCT) consolidates all the items, they can proceed to the Work Items List (WIL) refinement, based on the knowledge gained from the meeting so far. This activity is quite relevant because it will provide a Work Items List (WIL) that is ready to use in the Iteration Planning meeting. This means going over the following:

- **Work Items that did not meet the acceptance criteria** – Should they return to the top of the Work Items List (WIL), ready to be included and corrected in the next iteration or shall they be postponed for a future iteration?
- **New Items in the Work Items List (WIL)** – With the information provided by the Product Owner (PrOw) and the business stakeholders, the Agile Team Members (ATeM) shall agree (whenever possible) on relative estimates for each new added item. This will help the Product Owner (PrOw) to prioritise them in the Work Items List (WIL).

Depending on the number of new stories and the feedback received, the Agile Team Members (ATeM) may not be able to finish their analysis during this meeting. When they believe this will be the case, they should definitely focus on the items that, in theory, will be tackled in the next Iteration. All the rest can be handled after the Iteration Review, as part of the Release Planning exercise of the next iteration.

When the Iteration Review is finished (or a couple of hours after, if the Agile Team Members (ATeM) need to perform some estimates after the meeting), the Agile Project Core Team (A-PCT) must have a properly refined and prioritised Work Items List (WIL) which they feel comfortable to use as the basis for the planning exercise of the next iteration. The success of an Iteration Review is not solely linked to the number of Work Items approved. In fact, ensuring that the team collected solid and relevant feedback that will guide them towards the needs of the business stakeholders is, without any doubt, the best indicator of success.

### 5.4.3 Guidelines and Participants

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates - to the extent required by the Agile Team Members (ATeM) – several activities of the ceremony. She will ensure the previously agreed structure of the meeting is followed, as also the time boxes defined. She can guide the wrap-up activity by facilitating the discussion of each relevant Work Item. |
| Business Manager (BM) | Consulted about the demonstrated items and new/deleted work items. |
| Agile Team Member (ATeM) | Responsible for the setup and execution of this ceremony. She aligns expectations at the beginning. She can deliver the demo or delegate to the Product Owner (PrOw). In that case, she will support the Product Owner (PrOw) by helping the remaining stakeholders going through the demo. She will drive the wrap-up session and will deliver the relative estimates. |

| | |
|---|---|
| Product Owner (PrOw) | Accountable for the output of the meeting. She supports the execution of the ceremony by providing feedback regarding the demonstrated items (accept or reject according to the acceptance criteria) and helping with the setup of the ceremony when she conducts the demo. |
| Architecture Owner (ArOw) | Acts as an Agile Team Member (ATeM). |
| Business Implementation Group (BIG) | Consulted about the demonstrated items and new/deleted work items. |
| Project Manager (PM) | Informed about the outputs of the ceremony. |

***Guidelines***

- **Expose the Work Items to the Product Owner in advance** – During the iteration, the Agile Team Members (ATeM) should demonstrate every work item to the Product Owner (PrOw) as soon as it's ready. By ready doesn't mean necessarily perfect but good enough to allow the Product Owner (PrOw) to understand it and see if it's going in the right direction. By the time the work item is demonstrated in the Iteration Review, the Product Owner (PrOw) will be already familiarized with the item and consequently, more comfortable when validating the acceptance criteria.
- **Technical work should be demonstrated** – Every item that is part of the Work Items List (WIL) must be demonstrated to the Product Owner (PrOw). However, Agile Team Members (ATeM) tend to forget this rule when they have to build technical work items (enablement work), mostly because they are "invisible" and the acceptance criteria was defined by the Agile Team Members (ATeM). Because this kind of work plays a key role in the overall solution, it's very important that the Product Owner (PrOw) is not only aware of their existence but also understands its impact  in the overall solution. For the sake of transparency, PM²-Agile strongly recommends that those items are also demonstrated during the Iteration Review.
- **Keep the demo short and straight to the point** – A demo is not a playground to do random testing (this should be provided during the iteration when exposing work items for the first time). In a demo, the Agile Project Core Team's (A-PCT) goal is to confirm that a set of items are implemented properly so they can deliver an iteration goal. To achieve this, the demo must be focused only on the items agreed to validate and nothing else. This will keep the Product Owner (PrOw) and other stakeholders focused. Solid facilitation skills and the help of the entire Agile Project Core Team (A-PCT) is important to achieve this.
- **Build test cases targeting the acceptance criteria** - To enable short and straight to the point demos, one needs straight to the point test cases. This is achieved by developing test scenarios based on the acceptance criteria that must be validated. This ensure that the time taken during the demo is just the strictly necessary since audience will be focused on what is relevant.
- **Embrace change** - Several teams find it hard to accept changes as part of the review meeting process. Not only they are normal but they are actually welcome. That means that the business is engaged in finding the best solution and that is essential for the success of the project. If, on the other hand, the business stakeholders keep coming with requests that constantly change the direction of the work being done, that may mean a systemic problem, probably related either with a not so clear vision of the solution or the Product Owner (PrOw) not spending enough time with the team, etc. Whatever is the case, the Team Coordinator (TeCo) needs to step in and investigate.
- **Update the Release Plan** – After the Iteration Review is finished, new data related to the velocity of the Agile Project Core Team (A-PCT) will be generated. PM²-Agile recommends reflecting this data in the Release Plan, as it may impact the contents of what is being delivered in the next release. This updated should be done by the Team Coordinator (TeCo) and then communicated to the Product Owner (PrOw). However, if the team has an Iteration Retrospective after the Iteration Review, this exercise can then be done after, since the latest usually contributes with new Work Items to the Work Items List (WIL), which can have an impact in the Release Plan.

| **RAM** (RASCI) | **BIG** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|---|
| Iteration Review | C | C | I | **S** | **A/S** | **R** | **R** |

## 5.5 Iteration Retrospective

The idea of retrospecting is not new in PM². For instance, one of its key ceremonies - the Project-End Review meeting - aims to capture lessons learned and post-project recommendations to help teams benefiting from the experience acquired during the project. Moreover, PM² points the need to capture these improvement opportunities throughout the project and not just at the end, where most of them may have already been forgotten.

With PM²-Agile, there are two key elements that will take retrospection one step further. On one hand, the retrospective ceremony (Iteration Retrospective) happens frequently (by the end of each iteration), allowing a permanent and recurrent evaluation of the Agile Project Core Team's (A-PCT) performance made by the team itself. On the other hand, the Agile Project Core Team (A-PCT) will not wait by the end of the project to make adjustments. As soon as possible, the improvement opportunities are integrated into the team's work, allowing them to evolve and improve.

The Iteration Retrospective is part of the Reviewing stage of the CIR rhythm that supports PM²-Agile and it acts has the closure of the cycle. By the end of the project, when it's time to collect Lessons Learned, the repository where the Agile Project Core Team (A-PCT) keeps the logs of the Iteration retrospectives, is an invaluable source of information that will dramatically enrich the lessons learned and the post-project recommendations.

It's also important to mention that the Iteration Retrospective meeting is focused on real problems that affect a team. During this meeting, the Agile Project Core Team (A-PCT) finds solutions that can be implemented without having to wait for management's approval. Since these improvements are chosen by the Agile Project Core Team (A-PCT) and not imposed, the Agile Team Members (ATeM) are even more committed to their success.

There are several ways to organize and structure an Iteration Retrospective. PM²-Agile recommends a structure where its flexibility allows to easily adjust the duration and the activities used to reach the goal of each step of the meeting.

The following sections provide more details on the expected duration and the different steps of the Iteration Retrospective.

### 5.5.1 Frequency and Duration

The frequency of an Iteration Retrospective is aligned with the other ceremonies that aim to prepare and evaluate an iteration. Just like the Iteration Planning and the Iteration Review, the Iteration Retrospective happens once every iteration. Since the goal is to inspect what happened during the iteration and find opportunities for improvement, it's not a surprise that this ceremony takes place at the end of the iteration. This will allow the Agile Project Core Team (A-PCT) to collect as much data as possible and analyse it during the meeting to help generating important insights.

The duration of the Iteration Retrospective depends on several different factors.

- **Size of the team** – It seems natural that the more Agile Team Members (ATeM) participate in the Iteration Review, the more inputs will be collected, which means more time needed to analyse and process them.
- **Size of the Iteration** – With longer iterations, it's more likely to have a higher number of events or other topics the team wants do discuss, which means, in theory, more time is required.
- **Level of stress/friction** – Depending on the level of stress, controversy and other distressing aspects, more time must be put aside for the Agile Team Members (AteM) to vent.

While other factors may also impact the duration of the Iteration Retrospective, it's very important to ensure that the time allotted to each of the sections of the meeting is observed, otherwise the Retrospective may not generate the desired outputs. Still, PM²-Agile suggests a duration between 60 and 120 minutes depending on the factors described.

The next section provides detailed information about the structure of the Iteration Retrospective.

### 5.5.2 Structure of an Iteration Retrospective

The key goal of an Iteration Retrospective is helping teams to improve continuously. In order to make that happen, one needs to create an environment where all the Agile Team Members (ATeM) feel safe and comfortable sharing concrete data, identifying problems and improvement opportunities and establishing action plans. This environment must be created during every Iteration Retrospective and to enable that, a rigorous but flexible structure is required. Rigorous because it should always follow a set of steps and flexible to enable adjusting the contents of each step without changing its purpose.

More than just identifying what went well, what needs improvement and what needs to stop being done, PM²-Agile proposes a more agnostic structure[17], based on the following steps:

- **Welcome** – Establish an environment where everyone feels encouraged and comfortable participating and sharing opinions.
- **Collect Hard Data** – Collect all relevant data to help creating a common picture of the reality the team wants to focus on. This can include events, metrics, etc.
- **Understand Data and Generate Ideas** – Understand root causes of the problems and patterns of success. Generate possible solutions with several alternative scenarios.
- **Prioritise and Choose** – Because usually it's not possible to implement all the proposals, the team needs to prioritise and decide which ones to take onboard.
- **Closing** – End the ceremony and make sure to keep track of all the learnings.

The following sections describe a bit more in detail each of these steps.

#### *5.5.2.1 Welcome*

Although this step may sound a bit simple and not as relevant as the other ones, the fact is that this is the most important step to ensure a successful retrospective.

Every time the Agile Team Members (ATeM) get together to discuss topics that include their individual performance, it's very important that they feel comfortable, and they know that their time will be well spent.

The first key point is to make sure everyone feels that their opinion counts and as such, they will not have a problem sharing it. As an excellent and simple exercise, the Team Coordinator (TeCo) can ask each Agile Team Member (ATeM) to point out, using one sentence, how they felt during the iteration or what do they expect from the Iteration Retrospective. By doing this, the Team Coordinator (TeCo) makes it clear that she expects everyone to talk. When this exercise is not done or people are allowed to remain silent, it means that it's ok for them to keep silent for the rest of the retrospective. The Team Coordinator (TeCo) can use other activities to achieve this same goal.

After this warm-up, the Team Coordinator (TeCo) must explain the approach to the participants. What are the steps, the corresponding activities and what is expected from each. Finally, she must communicate the timebox for the ceremony.

The last point to consider in this step is to properly define the rules of the ceremony. These are the rules that will create the environment that allows the Agile Team Members (ATeM) to feel comfortable having difficult conversations. In other words, to define the accepted behaviours during the ceremony.

If the Agile Project Core Team (A-PCT) already has working agreements, they should be used. After all, the same principles apply to the Iteration Retrospective. If not, they should create them right now. This will take some time from this particular Iteration Retrospective but will be very useful for the following ones. The team is now ready to proceed to the next step.

#### *5.5.2.2 Collect Hard Data*

With the whole Agile Project Core Team (A-PCT) ready to start, it's time to collect all the data that will be analysed. The goal of this approach is to ensure that all Agile Team Members (ATeM) are focused and share the same vision of the reality. Otherwise, each one will have their opinions based on their own vision of the data they want to analyse.

As a starting point, the Team Coordinator (TeCo) should help the rest of the Agile Team Members (ATeM) collecting all the hard data related with events, metrics, completed work items, etc. Events may include all kinds of moments that were somehow relevant to any of the Agile Team Members (ATeM), like meetings, important decisions, celebrations, etc. Metrics can include burnup and burndown charts, velocity charts, percentage of broken builds, percentage of test coverage, etc. The more data the Agile Project Core Team (A-PCT) puts together, the easier will be to have a common view of the reality. Please note that although the

[17] Adapted from: *Agile Retrospectives – Making Good Teams Great*: Derby, Ether; Larsen, Diana – The Pragmatic Programmers

Team Coordinator (TeCo) can help with tasks, it's always the Agile Team Members (ATeM) who decide which data they want to use.

PM²-Agile suggests that the Team Coordinator (TeCo) and the rest of the Agile Team Members (ATeM) have this hard data visible, either organised by technical area, team area, chronologically in a timeline, etc. This approach will make it easier to visualise everything and make connections between the data, establish patterns, etc.

There's a second and very important aspect that cannot be left unattended by the Agile Team Members (ATeM) when analysing the data. It has to do with people's feelings. Feelings help understanding what is important to each Agile Team Member (ATeM) and how they felt in certain moments. By connecting the feelings to a set of data, a pattern, etc, it will become much easier to understand it. This may be finding the root cause of a problem, understanding a pattern, etc. This is a very important aspect as it will be a key enabler for the next step.

#### *5.5.2.3 Understand Data and Generate Ideas*

It's very important to have a set of hard data that can be analysed under a unique perspective by the several Agile Team Members (ATeM). However, having data is not enough, since data needs to have a meaning. That's when the Agile Project Core Team (A-PCT) should move to the next step. This third step focus on evaluating and understanding the data and generate ideas that will allow them to solve some of the problems identified, understand patterns, improve other aspects, etc. When a team implements a solution as an answer to a problem without considering other alternatives, usually the final result is not the best. For that reason, understanding the hard data and generating ideas plays a fundamental role, since this is what will allow the Agile Project Core Team (A-PCT) to improve its performance.

#### *5.5.2.4 Prioritise and Choose*

By this time, the Agile Project Core Team (A-PCT) should have a set of actions regarding possible improvements to implement. This is the right moment to prioritise and choose the ones the team wants to work on during the next iteration (or release). The Team Coordinator (TeCo) should guide the team towards the actions that will clearly bring the best benefits and which they can commit to.

Another important aspect is the way in which the actions chosen by the team will be planned in terms of work. To ensure that those actions are always visible and treated as any other item, PM²-Agile recommends including them in the Work Items List (WIL).

As a last point to consider in this step, it's important to ensure that for each of the work items planned (either during the Iteration Retrospective or during the Iteration Planning), Agile Team Members (ATeM) commit to execute all the required tasks. When a task is assigned to the Agile Project Core Team (A-PCT) and not to an Agile Team Member (ATeM), it ends up not being done.

#### *5.5.2.5 Closing*

An Iteration Retrospective can be very demanding from a psychologic point of view. As such, and just like the team takes a few minutes to prepare itself for the meeting, it should also take a few minutes to ensure a proper meeting closure.

The first aspect the Team Coordinator (TeCo) should focus on is to guarantee a decisive end. She must not allow pending topics and conversations to progress to this stage.

It's during this stage that the team decides how they will capture everything they discussed and learnt. Apart from what is registered in the Work Items List (WIL), there are several other topics whose information must be collected and stored. Boards, photos, printing digital images, etc. All this knowledge belongs to the Agile Team Members (ATeM) and not to the Team Coordinator (TeCo). They need to own it and store it.

Finally, the Team Coordinator must close the ceremony, acknowledging all the work made by the Agile Project Core Team (A-PCT), not only during the Iteration Retrospective but also throughout the entire Iteration. The Team Coordinator (TeCo) may also want to conduct a quick *Inspect and Adapt* towards the Retrospective to check which points can be improved.

### 5.5.3 Guidelines and Participants

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports/facilitates the Iteration Retrospective by selecting, preparing and facilitating all the required activities to be used in each of the different steps of the meeting. |
| Business Manager (BM) | Informed of the outputs of the retrospective if necessary. |
| Agile Team Member (ATeM) | Responsible for the execution of all the activities foreseen in this ceremony, including collecting all the hard data, understand it and define action plans.<br>Accountable for all the work items and activities that come as outputs of this ceremony. |
| Product Owner (PrOw) | Informed of the outputs of the retrospective if necessary. |
| Architecture Owner (ArOw) | Acts as an Agile Team Member (ATeM). |
| Project Manager (PM) | Informed about the outputs of the ceremony. |

***Guidelines***

- **Have working agreements defined**. – Working agreements are very important in the day-to-day activities of the Agile Project Core Team (A-PCT), including the several different ceremonies. This is especially true for Iteration Retrospective. Because of its format and contents, these agreements become even more important as they contribute decisively to a smooth ceremony.
- **Have a fully dedicated facilitator** – An Iteration Retrospective requires an active and focused presence of all Agile Team Members (ATeM). This is not compatible with the also active and fully dedicated presence of the facilitator. The Team Coordinator (TeCo) is the right facilitator.
- **Send the agenda previously to everyone** – Although the agenda is explained in the first part of the Iteration Retrospective, sending it in advance (activities included) allows all Agile Team Members (ATeM) to understand how they are going to invest their time and to prepare specific materials when needed.
- **Avoid the "no man's-land" responsibility** –Agile teams tend to create action items which they believe they are not responsible for by putting them on someone else's hands.
- By doing this, they will no longer be able to control how they can improve, making the retrospective a very limited exercise in terms of its main goal: The improvement of the Agile Project Core Team (A-PCT).
- **Record all the learnings, actions and people accountable** – One of the most relevant outcomes of each Retrospective is the increased knowledge that the Agile Project Core Team (A-PCT) gains. However, because memory cannot keep track of everything, the outputs must be recorded in a Retrospective Log (can be a document, a wiki, etc). This will be used as a reference in the next Iteration Retrospective but also by the end of the project, when it's time to collect the Lessons Learned. Please refer to the section 7.3.2 Project Logs for more information on Project Logs.

| **RAM** (RASCI) | BM | PM | TeCo | PrOw | ArOw | ATeM |
|---|---|---|---|---|---|---|
| Iteration Retrospective | I | I | S | I | R | R/A |

# 6 Roles and Responsibilities

The following diagram provides an overview of PM² project organisation.

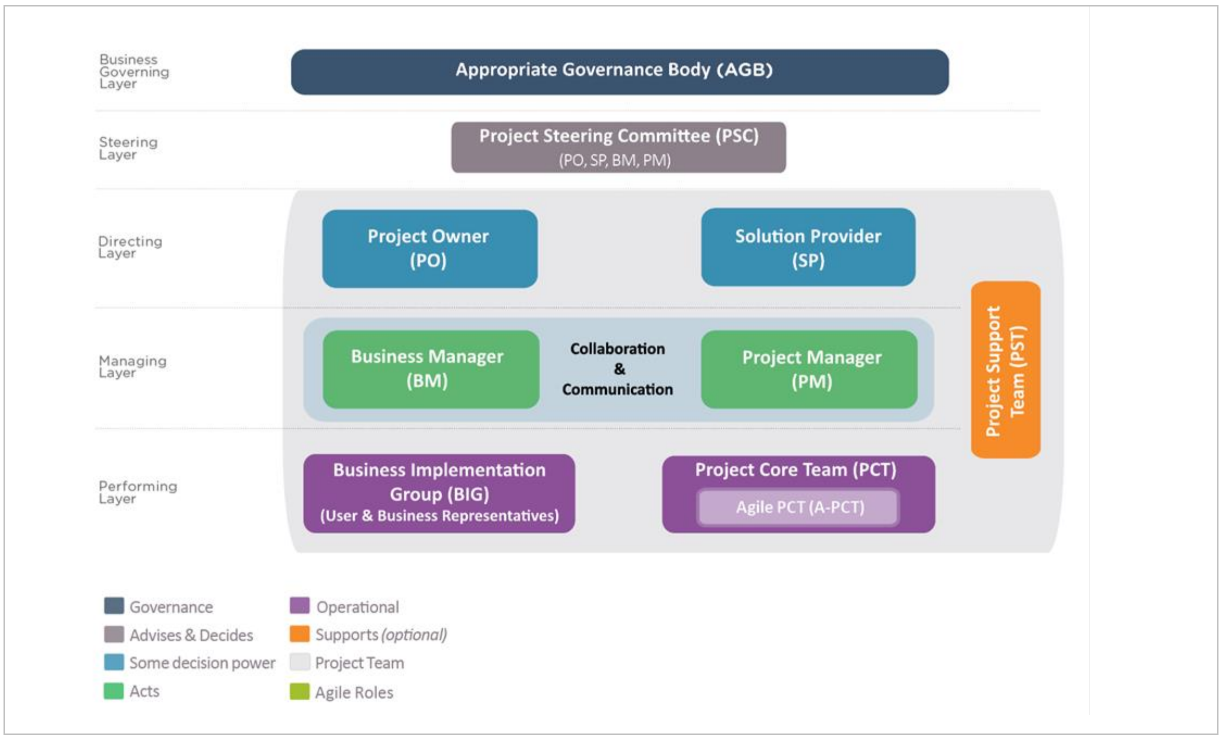


**Fig. 6.1** PM² Project Organisation

A short description of the main **roles** is provided in the table below:

| **Roles** | **Short description** |
|---|---|
| **Directing layer** | |
| Project Owner (PO) | The Project Owner (PO) is the key project decision maker and accountable for project success. |
| Solution Provider (SP) | Assumes overall accountability for the project deliverables. |
| **Managing Layer** | |
| Business Manager (BM) | Delegate of the Project Owner (PO) collaborates with the PM. |
| Project Manager (PM) | Is responsible for the whole project and its deliverables. |
| **Performing Layer** | |
| Project Core Team (PCT) | Plays a key role in project delivery. |
| Business Implementation Group (BIG) | Plans and implements the business change activities. |
| User Representatives (URs) | Represent the interests of the users in the project. |

PM²-Agile expands this project organisation in the Performing Layer. Based on the different roles and responsibilities involved in IT projects, the specific PM²-Agile roles and responsibilities are presented in the following sections.

Here are some useful reminders.

- There is one project team which includes all the Roles described in the Project Organisation.
- There is only one Project Core Team (PCT).
- A team applying PM²-Agile approach is referred to as the 'Agile PCT' (A-PCT).

## 6.1 PM²-Agile Project Organisation

- Team Coordinator (TeCo)
- Product Owner (PrOw)
- Architecture Owner (ArOw)
- Agile Team Member (ATeM)

Figure 6.2 shows the roles and responsibilities organising and managing the Project Core Team with the PM²-Agile approach.

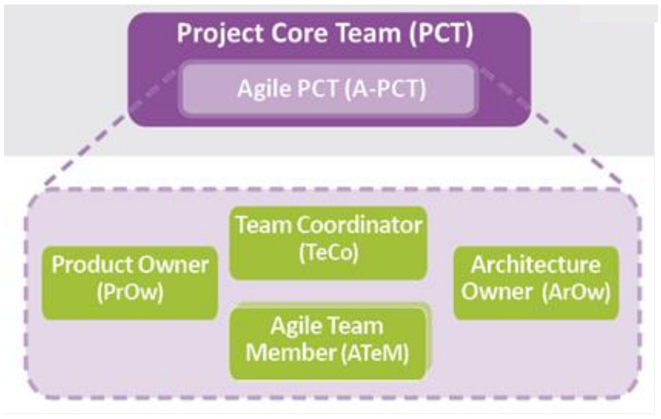


**Fig. 6.2** Main roles of the Agile Project Core Team (A-PCT)

## 6.2 Stakeholders

Project stakeholders are people (or groups) who can affect or can be affected by both the activities performed during the life of a project, or/and by the project's output and outcome. Stakeholders can be directly involved in a project's work or be members of other internal or external organisations (e.g. contractors, suppliers, users or the general public).

Depending on the complexity and scope of a project there may be few or many stakeholders. However, their effective involvement is a crucial success factor.

## 6.3 Core Roles

The following sections describe the Core Roles & Responsibilities of the Agile Project Core Team (A-PCT).

### 6.3.1 Team Coordinator (TeCo)

The Team Coordinator (TeCo) acts as a facilitator and team coach whose main purpose is to create and maintain the conditions (e.g. resources, issue resolution), allowing the team to achieve specific objectives and be successful. This role's responsibilities typically focus on the work's methodology aspects and hides the project management concerns from the Agile Project Core Team (A-PCT). The Team Coordinator (TeCo) facilitates or guides the team in performing technical management activities instead of taking on these responsibilities.

**Responsibilities**

- Helps the Agile Project Core Team (A-PCT) to continuously improve their way of working to become more effective as a team.
- Facilitates the collaborative and cooperative working environment within the Agile-Project Core Team (A-PCT).
- Facilitates the planning and estimation activities of the Agile Project Core Team (A-PCT).
- Responsible for reporting on the work progress (Development status report) to the Project Manager (PM).
- Ensures that the Agile Project Core Team (A-PCT) can be fully dedicated to delivery-related activities and on achieving the defined specific goals.
- Facilitates the decision making within the Agile Project Core Team (A-PCT).
- Works actively to identify and help with all obstacles preventing the team to achieve the iteration objectives.

### 6.3.2 Product Owner (PrOw)

The Product Owner (PrOw) represents mainly client and end-user concerns. This role should develop a deep understanding of the needs and desires of the stakeholders. This understanding allows the Product Owner (PrOw) to capture and set priorities for the Work Items. This role represents the 'single voice of the stakeholders' within the Agile Project Core Team (A-PCT) and should work, ideally, in the same physical environment as the rest of team.

**Responsibilities**

- Continuously prioritises the Work Items List (WIL) to be addressed by the Agile Project Core Team (A-PCT) in alignment with feedback from both the stakeholder community and the Agile Project Core Team (A-PCT).
- Clarifies domain-related questions that the Agile Project Core Team (A-PCT) may have or ensures that a channel with the relevant stakeholders is open for collaboration and clarification.
- Facilitates requirements gathering and modelling sessions.
- Ensures that the stakeholder community is represented.
- Facilitates the presentation of project's intermediate outputs to the stakeholder community (demos).
- Ensures that the stakeholders understand the benefits of the Agile approach followed by the Agile Project Core Team (A-PCT).

### 6.3.3 Architecture Owner (ArOw)

The Architecture Owner (ArOw) is the solution architect responsible for the Agile Project Core Team's (A-PCT) architecture decisions. The Architecture Owner facilitates the creation and evolution of the overall solution design and considers existing or planned investments made in other information systems/components.

Because architecture can generate critical project risks, one must ensure the team mitigates those risks. Although the Architecture Owner is typically the senior developer on the team (sometimes also referred to as the technical architect, software architect, or solution architect), it is not a hierarchical position to which other team members report. The Architecture Owner should have a strong technical background and a thorough understanding of the business domain and he/she is expected to sign off and deliver work like any other team member.

**Responsibilities**

- Guides the creation and evolution of the architecture of the information system.
- Favours a collaborative, team-based approach while avoiding dictating the architectural direction.
- Conducts the technical discussions but is not solely responsible for the architecture.
- Leads the initial architecture envisioning effort at the beginning of the project and supports the initial requirements envisioning effort (particularly when it comes to understanding and evolving the non-functional requirements for the information system), focusing on the project lifecycle and also on the evolution and maintainability of the information system.
- Ensures the alignment of the architecture of the information system with the guidelines and recommendations of the Architecture Office (AO) and the support of the established enterprise architecture principles.
- Leverages existing and/or planned IT investments in the organisation by continuously promoting a culture of reuse and interoperability within the Agile Project Core Team (A-PCT).
- Contributes to the organisation's set of reusable IT assets by considering the overall domain which the information system will support and the overall IT strategy.
- Informs the Team Coordinator (TeCo) and the Project Manager (PM) of the main architectural risks and contributes to defining a suitable risk management strategy.

### 6.3.4 Agile Team Member (ATeM)

The Agile Team Member (ATeM) focuses on producing the actual information system that is part of the project's solution to the stakeholders' needs. This role encompasses different information system development disciplines such as architecture, analysis, design, programming, testing, planning and estimation. An Agile Team Member (ATeM) has cross-disciplinary skills though the degree of specialisation in each discipline varies from individual to individual. Independent of the background and experience, the Agile Team Member (ATeM) collaborates with the rest of the Agile Project Core Team (A-PCT) members in a skill-growing environment. They can be seen as generalists as opposed more traditional, highly-specialised roles such as developers, analysts and testers.

**Responsibilities**

- Participates in planning and estimation of iterations, releases.
- Participates in the solution architecture design.
- Develops part of the information system, in collaboration with the solution architecture design.
- Tests developments.
- Provides progress information to the Team Coordinator.
- Decides the best way to do the work in each iteration.
- Communicates and collaborates with the rest of the Agile Project Core Team (A-PCT).

## 6.4 Other Roles

Apart from the core PM²-Agile Roles described so far, another set of roles may also be part of project's lifecycle. Typically, these other roles join the Agile Project Core Team (A-PCT) on a temporary basis and help the team address and overcome specific challenges, both from a business and technical perspective.

**Domain Expert**

The Product Owner (PrOw) is regarded as the 'single voice of the stakeholders'. However, inviting domain experts, such as business analysts for the business domain that the information system supports, helps the Agile Project Core Team (A-PCT) better understand the overall context and scope of the project.

**Technical Expert**

The Agile Project Core Team (A-PCT) comprises cross-disciplinary skilled individuals who can tackle most of the technical challenges during the project. Yet, for some projects, it might be useful to have technical experts' contribution, even if only on a temporary basis. Technical experts possess specialised knowledge and experience in a given software development area and help the Agile Project Core Team (A-PCT) address and overcome technical challenges. Specialised areas may include software architecture, user experience (UX), security, database administration, among others. For example, a technical expert from the Architecture Office can help the team develop a solution that is coherent with the overall enterprise architecture.

**Independent Tester**

Agile Project Core Teams (A-PCT) are responsible for the quality of an iteratively and incrementally delivered solution. Testing of the software is part of the responsibilities of all Agile Team Members (ATeM). Yet, some organisations require projects' software to be validated by a quality assurance team, working in parallel with the Agile Project Core Team (A-PCT).

## 6.5 The Project Manager (PM) and the Business Manager (BM) in a PM²-Agile Project

When introducing PM²-Agile within the PM² methodology, changes are expected regarding the governance model's roles. These changes reflect an approach that supports aspects like decentralised decision-making, self-organising teams, permanent business involvement and relentless improvement, just to name a few.

The introduction of new roles such as the Product Owner (PrOw) and the Team Coordinator (TeCo), and the acknowledgment of the development team as a "self-organising unit", impact PM² roles, such as the Project Manager (PM) and the Business Manager (BM). This impact reflects the need to align their responsibilities with the PM²-Agile principles.

### 6.5.1 Project Manager (PM)

In the context of PM², the Project Manager (PM) has several different responsibilities, which includes, among others, the following:

- Methodology compliance
- Team and progress management
- Managing stakeholders' expectations

The extended governance model defined by PM²-Agile identifies a set of new roles that will take on some responsibilities usually assigned to the Project Manager (PM).

**Methodology Compliance**

In the context of PM², one of the Project Manager's responsibilities is to actively encourage and educate the Project Core Team (PCT) on the proper usage of the methodology, as described in the Project Handbook.

When PM²-Agile is incorporated, this responsibility is shared with the Team Coordinator (TeCo), who will guarantee that all aspects related with agile values and principles are followed according to what is described in the Development Handbook. For example, the Team Coordinator (TeCo) is responsible for facilitating and assuring that the team properly conducts the agreed agile ceremonies (Iteration Planning, Review, Retrospective). The Project Manager (PM) participates in these ceremonies as a regular Agile Team Member (ATeM), while the Team Coordinator (TeCo) should be facilitating.

**Team and Progress Management**

PM² defines the Project Manager (PM) responsible for ensuring the efficient usage of resources. He/she assigns and monitors the work of each team member until the work is complete.

When Introducing PM²-Agile, the Agile principles of self-organising team and decentralised decision-making hold the Agile Project Core Team (A-PCT) responsible for managing its work (as described in the Development Work Plan). This approach allows the Project Manager (PM) to focus on the problems and impediments identified by the Agile Project Core Team (A-PCT) via the Team Coordinator (TeCo).

**Managing Business Stakeholders Expectations**

With PM²-Agile, the new Product Owner's (PrOw) role becomes the permanent point of contact between the requester and provider organisations. Regarding the scope of the solution and the corresponding time frame, he/she is responsible for aligning all expectations in collaboration with the Agile Project Core Team (A-PCT), the Business Manager (BM) and Business Implementation Group (BIG).

The Project Manager (PM), together with the Business Manager (BM), may need to intervene when the expectations are not aligned or the Product Owner (PrOw) and the rest of the Agile Project Core Team (A-PCT) cannot come to a consensus.

### 6.5.2 Business Manager (BM)

As defined in PM², the Business Manager plays a vital role in coordinating client-side activities and roles (e.g. user and business representatives) and ensures the project's deliverables fulfil the business and user needs. The adoption of PM²-Agile clearly extends the range of action of the Business Manager (BM) since the inclusion of the new Product Owner's (PrOw) role creates a permanent and active "client's voice" on the Provider's side.

**Liaison between User Representatives (URs) and the Provider Organisation**

The Product Owner (PrOw) drives the implementation of the solution based on a prioritised Work Items List (WIL). The prioritised WIL reflects the strategy and the needs of the client organisation and is the outcome of the regular work performed by the User Representatives (URs) and the Product Owner (PrOw) during refinement sessions.

Additionally, the Product Owner's (PrOw) ensures User Representatives (URs) participate actively in the Iteration Review to validate the development team's iteration output. These recurrent activities make the Product Owner (PrOw) the right person to act as a liaison between the User Representatives (URs) and the Provider organisation when PM²-Agile is adopted.

**Empowerment and Alignment with the Product Owner (PrOw)**

Because the Product Owner (PrOw) is part of the Agile Project Core Team (A-PCT), a more dynamic and expedited interaction between the "business" and "IT" emerges. Since the Product Owner (PrOw) embodies the business strategy and vision for the product being built, he/she must be:

- **empowered** by the Business Manager (BM) to take the necessary day-to-day decisions that allow the development team to progress towards the final product without having to wait for specific decisions;
- **aligned** with the Business Manager (BM) and the Business Implementation Group (BIG) to ensure the Work Items List reflects the business strategy and vision so that the team consistently and regularly focus on the most critical things.

An empowered Product Owner aligned with the Business Manager (BM) and the Business Implementation Group (BIG) allows the real benefits of PM²-Agile to surface.

## 6.6 RAM (RASCI) – Documenting Responsibilities Assignments

RASCI (pronounced 'Rass-Key') is also known as the Responsibility Assignment Matrix (RAM) and is a way of representing the responsibilities for the roles involved in specific (management) activities. All stakeholders should be aware of their role and responsibilities.

RASCI stands for:

| **RASCI** | | **Description** |
|---|---|---|
| **R** | **R**esponsible | They **do the work** and also lead others that **s**upport (also do work). |
| **A** | **A**ccountable | They **authorise and approve work**. Only **one** person is **a**ccountable. |
| **S** | **S**upports | As part of a team, they work with the person responsible. Unlike roles with **C**onsulted responsibility, the roles with **S**upport responsibility **help to complete the task**. |
| **C** | **C**onsulted | They **provide input** to the activity. |
| **I** | **I**nformed | They **will be informed** (kept up to date). |

This guide includes a RAM (RASCI) table for each of the artefacts presented in *Section* 7 *Artefacts*.

An example of the RAM for the roles involved in creating the Development Handbook is shown below.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Development Handbook | I | **A** | **R** | S | S | S |

- **Accountable**: The  Project Manager (PM) is accountable (ensures that the work is done).
- **Responsible**: The Team Coordinator (TeCo) is responsible for creating the Development Handbook.
- **Supports**: The Product Owner (PrOw) and the Architecture Owner (ArOw) work with the Agile Team Members (ATeM) to develop the Development Handbook. The final responsibility, however, lies within the hands of the Team Coordinator (TeCo).
- **Informed**: The Business Manager (BM) will be informed when the Document is complete.

# 7 Artefacts

Documenting the work planned and performed in the Agile layer is critical in increasing transparency and coordination between the different layers of the PM² project organisation (i.e. between the directing, managing and performing Agile layers).

Supporting PM²-Agile is a set of artefacts used to capture and document information regarding the management approach and Agile Project Core Team (A-PCT) activities, milestones, issues and progress reporting.

These artefacts are grouped in three categories: Agile-specific artefacts, Coordination and Reporting artefacts, and IT governance artefacts (common to all IT projects, regardless of the management approach chosen).

| Artefact type | Description |
|---|---|
| IT governance artefacts | Provide the information requested by the Organisation´s IT Governance. They consist of the Business Case, Project Charter, Architecture Overview, and Operational Model documents. |
| Agile-specific artefacts | Capture information for planning specific Agile processes, activities, releases, iterations and other milestones. |
| Coordination & reporting Artefacts | Capture the information needed to coordinate between overall project activities and those undertaken by the Agile Project Core Team (A-PCT) giving the Project Manager (PM) visibility of the domain-specific activities, issues, milestones and progress. |

Artefacts that support project work based on PM²-Agile.

- Business Case and Project Charter
- Architecture Overview
- Operational Model
- Development Handbook (becomes part of the overall Project Handbook)
- Development Work Plan (becomes part of the overall Project Work Plan)
- PM²-Agile Logs
- Deployment Plan
- Test Plans
- Development Status Report (becomes part of the Project Status Reports)

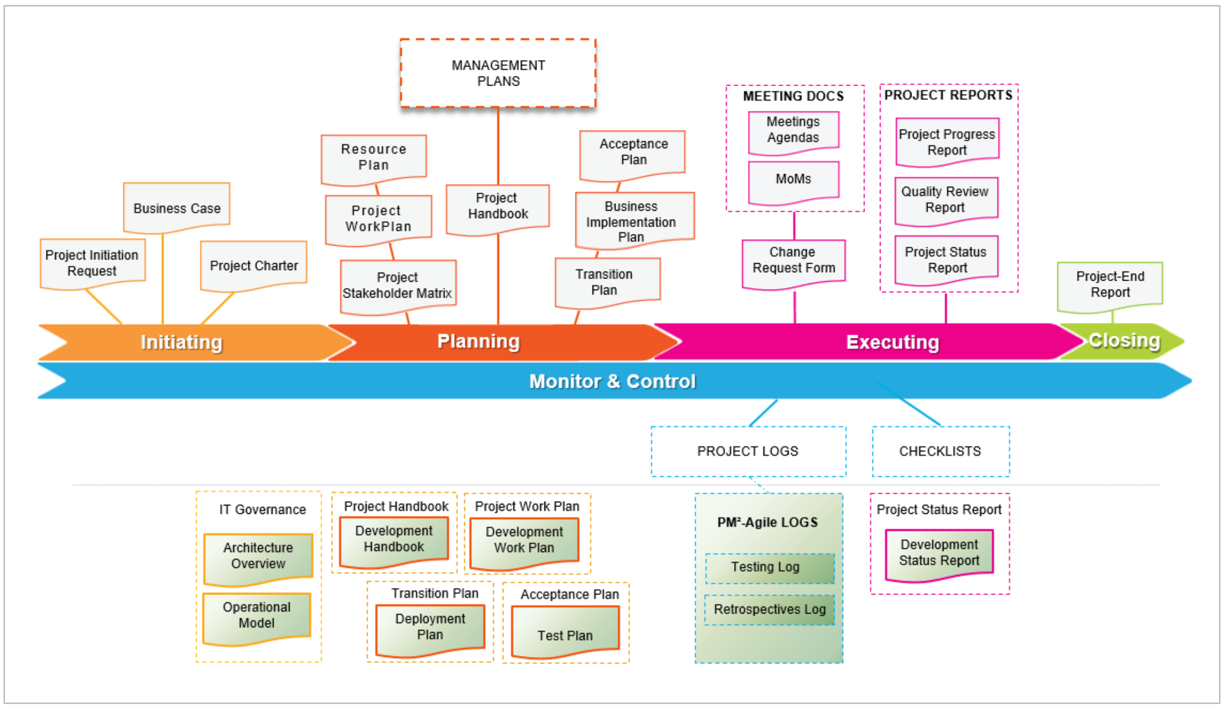


**Fig. 7.1** PM² and PM²-Agile (combined) Artefacts Landscape

## 7.1 IT Governance Artefacts

The existence of an Appropriate Governance Body (AGB) is a very important aspect that is foreseen in the organic structure of a PM² project. While evaluating and governing projects, a critical goal is to ensure that an outline of the proposed project's goals, scope, estimated budget, roadmap, risks, high-level deliverables, technical solution's core architectural elements and hosting requirements are provided before the project receives budget approval.

Within the Initiating phase, the IT governance body proposes a minimum set of documentation consisting of a Business Case, a Project Charter, an Architecture Overview and an Operational Model.

This set of IT governance documents aims to improve the information technology governance within the Organisation, balancing and leveraging its current and planned IT investments.

### 7.1.1 Business Case

The purpose of the Business Case is to capture project's rationale, describe alignment with the organisation's strategic objectives, justify the investment in time and effort and establish budgetary needs. For larger strategic projects, the Business Case may also assess impact and risks and provide a more detailed cost-benefit analysis.

The Business Case provides decision-makers with the information they need to determine whether the project is worth doing. The Business Case is a living document and therefore should be re-examined at critical project milestones to check that the expected benefits are still achievable, the costs/schedule fall within the budget/timeline, and the project is still relevant to the organisation and should be continued.

More information about the Business Case document can be found on the PM² Portal and guide. The table below and figure 7.1.2 provide a summary of the roles involved when creating this artefact.

| **RAM** (RASCI) | **AGB** | **PSC** | **PO** | **BM** | **UR** | **SP** | **PM** | **PCT** |
|---|---|---|---|---|---|---|---|---|
| Business Case | I | C | **A** | **R** | C | **S** | **S** | - |

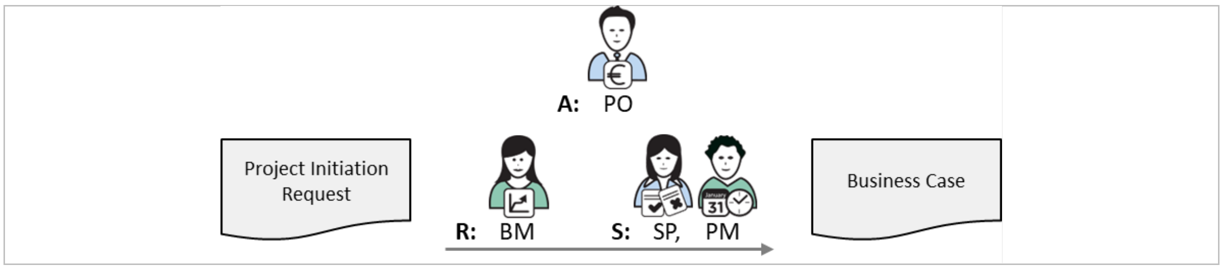


**Fig.7.2** Business Case — Inputs and main roles

### 7.1.2 Project Charter

The Project Charter provides a basis for the more detailed project planning. It defines the project's objectives (i.e. scope, time, cost, quality), high-level requirements, risks and constraints, milestones and deliverable(s).

The charter, a vital element of the project approval process, includes the what, how and when fundamentals of the project and provides a baseline against which progress can be measured. Although the Project Charter can be initiated by the Business Manager (BM), it is ultimately the responsibility of the Project Manager (PM) to complete it and submit it for approval.

More information about the Project Charter document can be found on the PM² Portal and guide. The table below and figure 7.3 provide a summary of the roles involved in the creation of this artefact.

| **RAM** (RASCI) | **AGB** | **PSC** | **PO** | **BM** | **UR** | **SP** | **PM** | **PCT** |
|---|---|---|---|---|---|---|---|---|
| Project Charter | I | **A** | C | **S** | C | **S** | **R** | C |

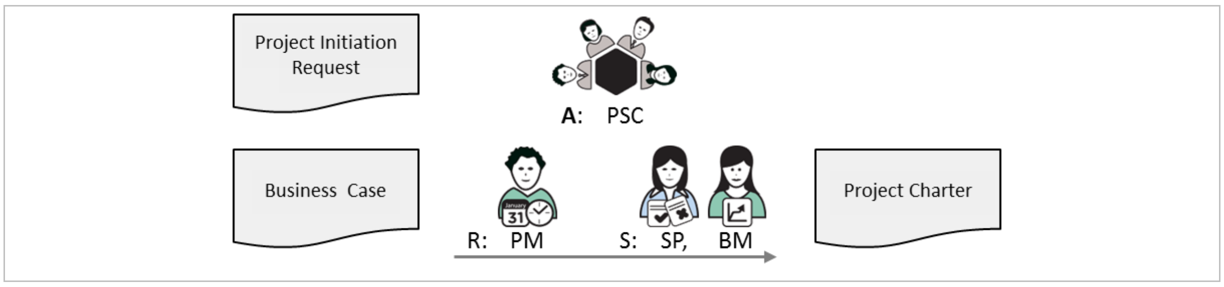


**Fig. 7.3** Project Charter — Inputs and main roles

### 7.1.3 Architecture Overview

The Architecture Overview provides a high-level view of the information system's software architecture to be developed, including the major building blocks of the architecture and a description of how the architecture contributes to all capabilities of the system.

The Architecture Overview formally documents the critical architectural decisions made and how compliance concerns (e.g. security, data protection, anti-fraud) are addressed by the information system. When an IT Reference Architecture is in place, the Architecture Overview documents how the information system uses the Reference Architecture guidelines to ensure consistency and efficiency throughout the IT landscape. The Architecture Overview also addresses reusability concerns by providing a checklist of commonly used components/services within the Organisation.

A key aspect of the Architecture Overview is to enable communication between the project stakeholders to understand the information system's structure and validate the implications of the chosen architectural approach. This artefact is also crucial when there are dependencies between information systems since it conveys, from a high-level perspective, what building blocks constitute the information system and what type of interfaces may exist to support interoperability.

The Architecture Overview is extremely important when developing solutions based on new approaches or technologies. In systems where there is already a well-defined architecture, this artefact can be lighter and refer to the existing architecture.

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports the Architecture Owner (ArOw) by facilitating the process of obtaining any needed information from the project level. |
| Agile Team Member (ATeM) | Supports and actively contributes to the definition of the Architecture Overview. |
| Architecture Owner (ArOw) | Responsible for creating the Architecture Overview and ensuring compliance with the organisation's standards regarding the IT Architecture (Reference Architecture). |
| Product Owner (PrOw) | Is informed of the Architecture Overview to have an overall understanding of the key architectural decisions and constraints. |
| Project Manager (PM) | Accountable for the Architecture Overview's output, the Project Manager (PM) needs to understand the essential architectural decisions and constraints and ensure that the project stakeholders are aware of them. |

**Guidelines**

- **Understand the Problem to be solved** – Before defining a solution and the corresponding Architecture, it is paramount to clearly understand what problem is to be solved and which needs are to be fulfilled. The Business Case and Project Charter provide this information.
- **Use both internal and external standards** – In the IT landscape, dozens of internationally recognised architecture standards and architectural patterns can be used to help making decisions and define software architecture. Additionally, the organisation itself may also have its own standards for which the Agile Project Core Team (A-PCT) and in particular the Architecture Owner (ArOw) should be fully familiarised.

- **Refer to the existing components' catalogue** – If there is a components' catalogue that an Architecture Office maintains, evaluate usage while ensuring that the trade-off in cost, risk and functionality is understood and accepted by the project stakeholders.
- **Make use of different models** – Because there are many diverse perspectives for Architecture, PM$^2$-Agile suggests using several different UML models (Class and Package diagrams, Object diagrams, Component diagrams, etc).
- **Take into account compliance concerns** – Every organisation has standards and policies that can affect the way the solution architecture is conceived. Therefore, compliance concerns need to be addressed in information system's architecture, especially in regards to security, document management and data protection.
- **Strive for a solid Architecture** - Have an architecture that can be (the following list is not exhaustive):
    - Consistent — All parts of the architecture are in harmony, and there are common approaches to partitioning and allocation.
    - Loosely coupled — Architectural components are independent and loosely coupled.
    - Generic — Architectural components function in various environments and contexts.
    - Extensible — The architecture can be extended.
    - Simple — A good architecture should be 'as simple as possible but no simpler.'
- **High level with just enough detail** - Ensure that the Architecture Overview, in an initial stage, is detailed enough to provide the overall technical approach at a high level.
- **Keep updating the Architecture Overview** - As the project evolves and there is a better understanding of the solution, refine the Architecture Overview to provide adequate support to the information system's development. As the development of the information system 'tests' the architecture, ensure that any decisions or changes are included in the Architecture Overview.
- **Supported by the Operational Model** - Ensure the alignment between the Architecture Overview and the Operational Model so that the 'operational' aspect of the IT system's architecture is up to date and hosting requirements are identified and managed.
- **Source of enablement work** – Because the Architecture Overview includes a picture of the architecture supporting the solution, It serves as an invaluable source of Enablers that will populate the Work Items List (WIL).

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Architecture Overview | I | **A** | **S** | I | **R** | **S** |

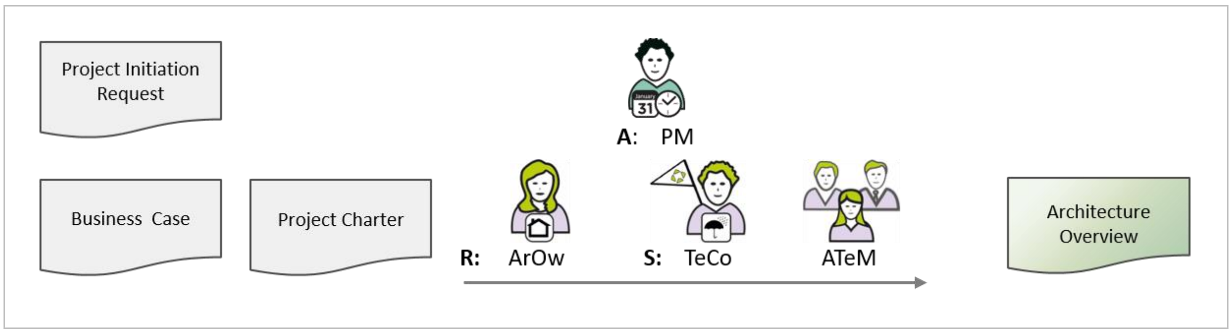


**Fig. 7.4** Architecture Overview — Inputs and main roles

| **PM² Lifecycle Phases:** | **Initiating** | **Planning** | **Executing** | **Closing** |
|---|---|---|---|---|
| Architecture Overview | *Created* | *Updated* | *Updated* | - |

### 7.1.4 Operational Model

The Operational Model describes the 'operational' aspect of an IT system's architecture, focusing on the system's infrastructure, both from a logical and physical perspective. It describes the required operational characteristics and the architecture's network of computer systems, associated peripherals, software, middleware, and application software used to support the system's users.

The operational distribution, which may be grouped into deployment units of a system's components, should be defined. Nodes, the placement of nodes and users across locations, and the connections between nodes necessary to support the required interactions between components should also be detailed to achieve the system's functional and non-functional requirements.

For IT projects hosted in a third party's Data Centre, the Operational Model is a key input since it will be extremely important for the system's proper support and maintenance.

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports by facilitating the process of obtaining any need of information from the Managing Layer. |
| Agile Team Member (ATeM) | Supports and actively contributes to the definition of the Operational Model. |
| Architecture Owner (ArOw) | Responsible for preparing the Operational Model and for ensuring compliance with the organisation's standards regarding the IT Architecture (Reference Architecture), hosting guidelines, etc. |
| Product Owner (PrOw) | Is informed to have an overall understanding of the key architectural decisions and constraints, as also the IT system infrastructure, both from a logical and physical perspective. |
| Project Manager (PM) | Accountable for the output of the Operational Model, the Project Manager understands and relays the key aspects of the needed IT system infrastructure and establishes a communication channel with the hosting services. |

**Guidelines**

- **Understand the Problem to be solved** - Ensure the project context and the proposed solution are understood by checking the Business Case and the Project Charter.
- **Keep updating the Operational Model** - As the project evolves and there is a better understanding of the solution, refine the Operational Model to detail the required operational characteristics of the IT system. It should describe, at an architectural level, the network of computer systems and associated peripherals, together with the systems software, middleware, and application software required to support the users of the system, both from a logical and physical perspective.
- **Make use of different models** – Because the IT infrastructure can be seen from different perspectives, PM²-Agile suggests using several UML models (such as Deployment diagrams).
- **Take into account compliance concerns** – Similar to the Architecture Overview, organisation standards and policies can impact the solution's infrastructure design. Ensure that compliance concerns are addressed in the information system's architecture, such as security, document management, data protection, etc.
- **Manage the feasibility of the Operational Model** - Manage the system's operational complexity and ensure that it is viable, both at the development and operational stage. This is why keeping alignment between the Architecture Overview and the Operational Model is such an important activity.
- **Everyone is involved** - Ensure the involvement of the whole Agile Project Core Team (A-PCT) and the broader project stakeholder community is informed of crucial operational decisions that may impact the solution.

| **RAM** (RASCI) | PM | TeCo | PrOw | ArOw | ATeM |
|---|---|---|---|---|---|
| Operational Model | **A** | **S** | I | **R** | **S** |

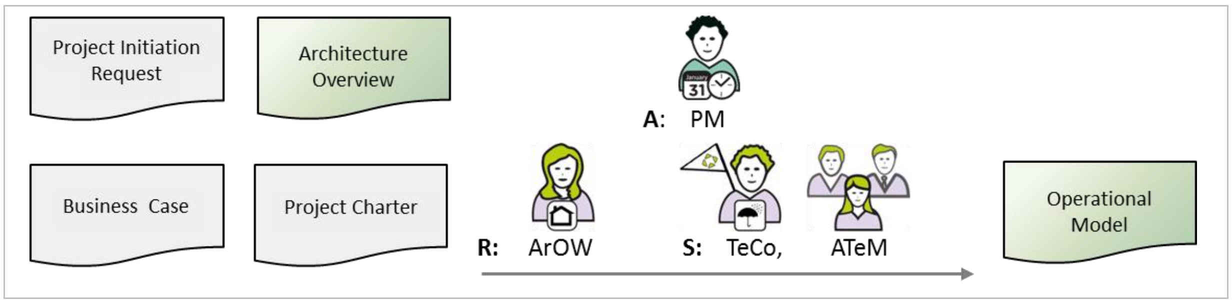


**Fig. 7.5** Operational Model — Inputs and main roles

| PM² Lifecycle Phases: | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Operational Model | *Created* | *Updated* | *Updated* | - |

## 7.2 Agile-specific Artefacts

PM²-Agile specific artefacts are produced by a self-organised Agile Project Core Team (A-PCT) where everybody is equally responsible. The coordination role for each artefact is very important but can be taken by any team member. However, there are default roles and responsibilities specified for each artefact. These are highly recommended in order to ease the coordination with other teams and the Managing Layer.

### 7.2.1 Development Handbook

The Development Handbook documents the selected approach for developing the information system (as part of the overall solution) and how the domain-specific aspect of the project relates to the overall project level. The Development Handbook is an extension of the Project Handbook and should be aligned with the key controlling processes, project policies, rules and overall management approach described. The Development Handbook is identified as a separate artefact but does not have to be a separate information carrier. Depending on the project's nature, there might be a single container for both artefacts and even a synchronised configuration management.

The Development Handbook extends the following areas of the Project Handbook.

- **Roles & responsibilities** – Documents any deviations from the standard PM²-Agile Roles and the standard RASCI table of responsibilities.
- **Project management plans** – Ensures that important information related to the Agile development aspect of the project is considered and documented in the PM² management plans, such as the selected approach to managing risks that are domain-specific but that may need to be visible and managed at the project level.
- **Domain-specific artefacts** – Identifies which Agile-specific artefacts will be used and documents any tailoring considerations (e.g. tools used to implement the Development Work Plan).

The Development Handbook and the Development Work Plan, together, form the basis for managing the project's development activities and are essential references for all project members and stakeholders, especially for the Agile Project Core Team (A-PCT).

Therefore, the Development Handbook is a living document. Any changes or improvements suggested as the outcome of ceremonies like Iteration Retrospectives or Iteration Reviews should be documented there. During the closing phase, the Development Handbook also becomes an essential point of reference for the project-end review meeting and should be properly closed and archived.

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Responsible for drafting the Development Handbook and ensuring alignment with the Project Handbook, in close collaboration with the Project Manager (PM). |
| Architecture Owner (ArOw) | Supports the Team Coordinator (TeCo), providing technical advice and identifying technical constraints that may impact the approach to be defined. |

| | |
|---|---|
| Product Owner (PrOw) | Supports the Team Coordinator (TeCo), providing insight and information related to the requestor's organisation reality that may impact the approach to be defined. |
| Agile Team Member (ATeM) | Supports the Team Coordinator (TeCo), providing additional insights on the tools and techniques that the team plans to use to support the solution's development. |
| Project Manager (PM) | As accountable, the Project Manager (PM) needs to be aware on how the Agile Project Core Team (A-PCT) plans to manage the development aspect of the project and ensure that the relevant information for the Project Handbook is captured from the Development Handbook. |

**Guidelines**

- **Use the Project Handbook as starting point** – For a PM² project, the Project Handbook is a reference and starting point for creating the Development Handbook. Decide if the Development handbook will be an information carrier on its own or if it will be embedded in the Project Handbook.
- **Adjust the Project Lifecycle accordingly** - PM²-Agile introduces the CIR (**C**oordinate, **I**mplement, **R**eview) rhythm. With that, frequent releases, Iterations, allow the standard phases boundaries to overlap and be more flexible.
- **Describe the Project Progress Measurement clearly** – Because Agile planning varies in its essence from more traditional approaches, using PM²-Agile implies a different way to measure progress as opposed to a standard PM² project. This must be described properly, including references to relevant tools like Agile Estimating.
- **Pay attention to core Project Process changes** – With PM²-Agile, part of the planning exercise relies on a completely different Requirements Management approach when compared to the standard PM² methodology. The use of stories as the working units that drive progress needs to be clear to all stakeholders. Furthermore, because Agile foresees "stakeholder collaboration over contract negotiation", the Change Management process must be adjusted to reflect this agility. Other processes like Resource Management and Communications Management should also be adjusted accordingly.
- **Include new PM²-Agile roles and responsibilities** – With PM²-Agile, the Agile Project Core Team (A-PCT) becomes part of the project and with it, a new set of roles and responsibilities is added to the PM² project organisation. Some of these roles will be closely related with other standard roles, reorganising some of their existing responsibilities. These changes must be very clear in the Development Handbook.
- **It's a living Artefact** – As part of the CIR rhythm, the Review aspect will bring frequent changes to the project's organics, regardless if it's process, team or solution related. As such, those changes must be reflected in the Development Handbook.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Development Handbook | I | **A** | **R** | **S** | **S** | **S** |

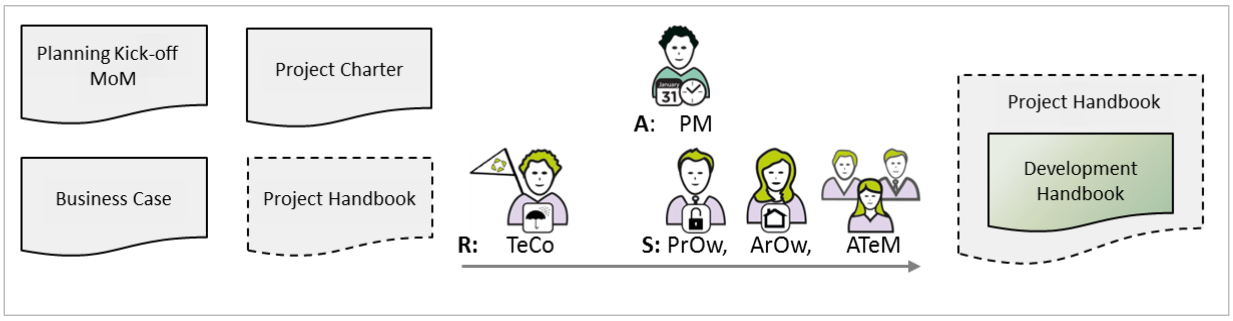


**Fig. 7.6** Development Handbook — Inputs and main roles

| PM² Lifecycle Phases: | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Development Handbook | - | *Created* | *Updated* | - |

### 7.2.2 Development Work Plan

The Development Work Plan gathers the relevant information that helps to manage the development aspect of the project. Viewed as a container of different perspectives on what needs to be done and how the development goals will be achieved. It includes the Work Items List (WIL), the corresponding Release Plan and the supporting iterations. These tree elements allow the Agile Project Core Team (A-PCT) to set and manage the other project organisation members' expectations.

The Development Work Plan is a living artefact that must be kept current by incorporating new information details and work progress status as they become available. It should be updated as often as necessary to capture and incorporate any changing business priorities and needs, including changes in the Work Items List (WIL) or in the contents of the Release Plan.

The Development Work Plan becomes part of the Project Work Plan and therefore should be aligned with important project milestones as defined by the Project Manager (PM) and the whole Project Core Team (PCT).

The Development Work Plan can be documented using varied formats and media (e.g. MS Office-based documentation, applications such as Atlassian JIRA, IBM Rational Team Concert, etc).

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Responsible for collecting data of all activities involved in the definition of the Development Work Plan in general, and for the Release Plan, in particular. |
| Agile Team Member (ATeM) | Actively supports the definition of the Development Work Plan, playing a critical role in the definition of the Iteration Plans. |
| Product Owner (PrOw) | The Product Owner also supports the definition of the Development Work Plan as a whole. Responsible for the Work Items List (WIL) and an important contributor for the definition of the Release and Iteration plans. |
| Architecture Owner (ArOw) | Supports the Team Coordinator (TeCo) providing technical advice and identifying technical constraints In the context of each iteration. Helps the Product Owner (PrOw) prioritising the Work Items List (WIL) and organising the release plans. |
| Project Manager (PM) | As accountable, needs to be aware on how the Agile Project Core Team (A-PCT) plans to deliver the solution and how will that be aligned with the overall project milestones. |

**Guidelines**

The Development Work Plan consists of two parts:

- **Work Items List** –The Work Items List includes all the items (e.g. user stories, bugs, enabler stories) that need to be addressed within the IT-domain-specific aspect of the project and proposed work that may affect the information system in this or future projects. All the Work Items need to be prioritised and estimated at some point, since the Work Items List (WIL) is the main driver of the project.
- **Schedules** – There are two schedule detail levels: a release level schedule (Release Plan) and an iteration level schedule (Iteration Plan). In the Release Plan, the focus is primarily on *what* needs to be done and *when*. In an Iteration Plan, the focus is on *how* it will be done by the Agile Project Core Team (A-PCT).

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Development Work Plan | I | **A** | **R** | **S** | **S** | **S** |

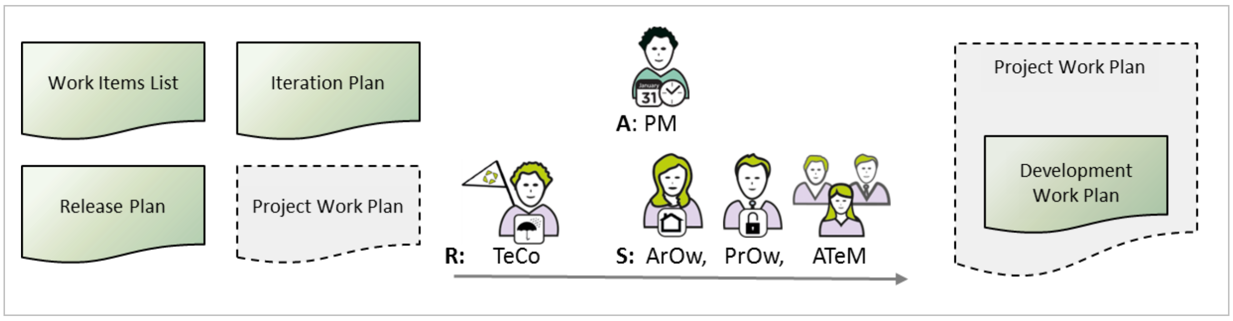


**Fig. 7.7** Development Work Plan — Inputs and main roles

| **PM² Lifecycle Phases:** | **Initiating** | **Planning** | **Executing** | **Closing** |
|---|---|---|---|---|
| Development Work Plan | - | *Created* | *Updated* | - |

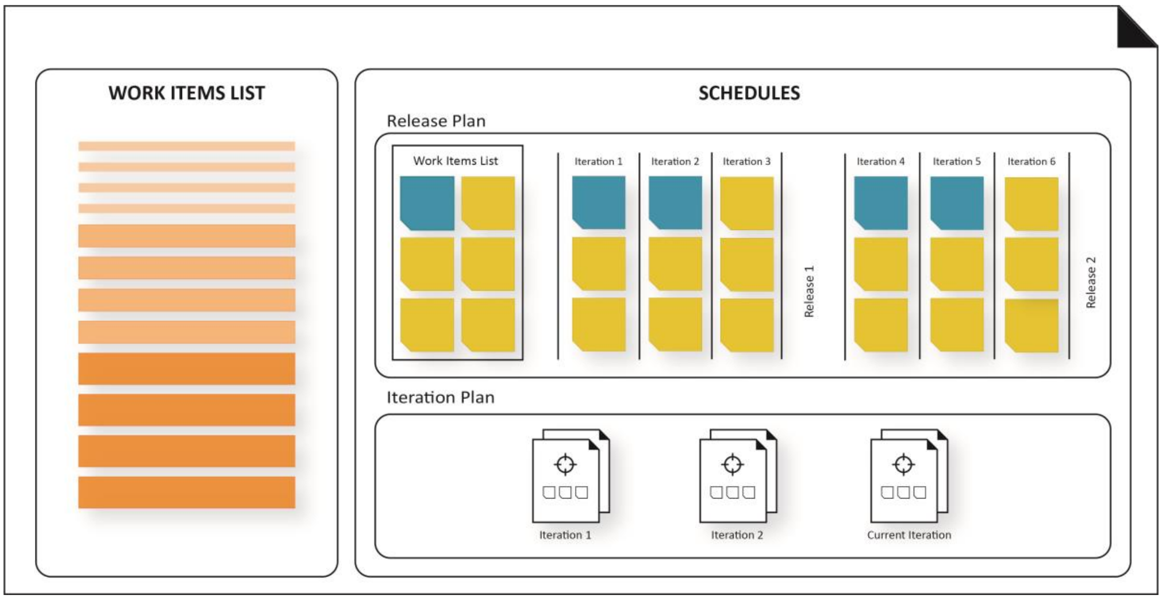


**Fig. 7.8** Development Work Plan

### *7.2.2.1 Work Items List*

As part of the Development Work Plan, the Work Items List (WIL) is a list of the work to be done during a project. Work items are intended to be prioritised, effort-estimated and tracked in terms of progress.

The following are typically included in the Work Items List (WIL):

- Defects (or bugs)
- Enabler Stories
- User stories

The Work Items List (WIL) is continuously maintained by the Product Owner (PrOw) throughout the entire project. During each iteration, the Product Owner (PrOw), the Business Implementation Group (BIG) and the Business Manager (BM) revisit the Work Items List (WIL) to ensure that it reflects the relevant work items to be implemented. The outcome of this exercise is then discussed with the rest of the Agile Project Core Team (A-PCT) so that some relative estimates can be defined and dependencies identified, helping the Product Owner prioritising the Work Items List (WIL).

A Work Item is not a contract between the development team and the Product Owner (PrOw) but rather a pointer to something that needs to be implemented. Therefore, each Work Item commonly refers to additional relevant information (business rules, diagrams, user interface specifications, etc.) to carry out the work.

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Consulted by the team and the Product Owner (PrOw) to guarantee the consistency and form of the Work Items List (WIL) that will allow the Agile Project Core Team (A-PCT) to work on it. Consulted by the Project Manager (PM) to verify alignment with the Project Charter. |
| Agile Team Member (ATeM) | Supports the Product Owner (PrOw) by providing inputs regarding estimates, dependencies, etc., that may impact the Work Items List organisation and prioritisation. |
| Architecture Owner (ArOw) | Supports the Product Owner (PrOw) by providing inputs regarding enabler work items that need to be delivered to support business work items. |
| Business Manager | As a representative of the Project Owner on the daily activities of the project, supports the Product Owner (PrOw) by giving inputs regarding the business priorities, objectives and user needs. |
| Product Owner (PrOw) | Responsible for the definition and maintenance of the Work Items List (WIL), fully aligned with the Business Manager (BM) and empowered to make all the necessary decisions on the daily life of the Agile Project Core Team (A-PCT). |
| Project Manager (PM) | As accountable, he needs to be aware on how the Work Items List (WIL) (as part of the Development Work Plan) is aligned with the scope defined in the Project Charter. |

**Guidelines**

- **Project Charter as a generator of Work items** – As one of the first things to be done when creating the Development Work Plan, the Features' list defined in the Project Charter will be the key generator of the work items that will populate the Work Items List (WIL).
- **Architecture Overview and Operational Model** – These two artefacts are an important source of enabler stories that will support the implementation of the "business part" of the solution.
- **Start working with a prioritised Work Items List (WIL)** – The Work Items List (WIL) doesn't have to be complete for the Agile Project Core Team (A-PCT) start working, but it must be prioritised. Without a prioritised Work Items List (WIL), the team runs the risk of investing resources in not so important items.
- **Product Owner (PrOw) maintains the Work Items List (WIL)** – The Product Owner's (PrOw) responsibility is to maintain the Work Items List (WIL) updated and prioritised, ensuring the Team is always working on the most relevant Work Items.
- **Refer to Release Planning, Iteration Review and Iteration Retrospective** – As part of the continuous learning process and because the context of the solution will change, it's very important to refer to ceremonies like Release Planning, Iteration Review and Iteration Retrospective as they continuously provide inputs and generate work items to be included in the Work Items List (WIL).
- **Project Logs as a source of work items –** Managing the project logs (Change requests, Risks, Issues and Decisions) can generate several work items that must be included in the Work Items List (WIL). A Change request may generate several new stories to be developed to deliver the change request or the plan to mitigate an architectural risk my include implementing some enabler stories.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Work Items List | **S** | **A** | C | **R** | **S** | **S** |

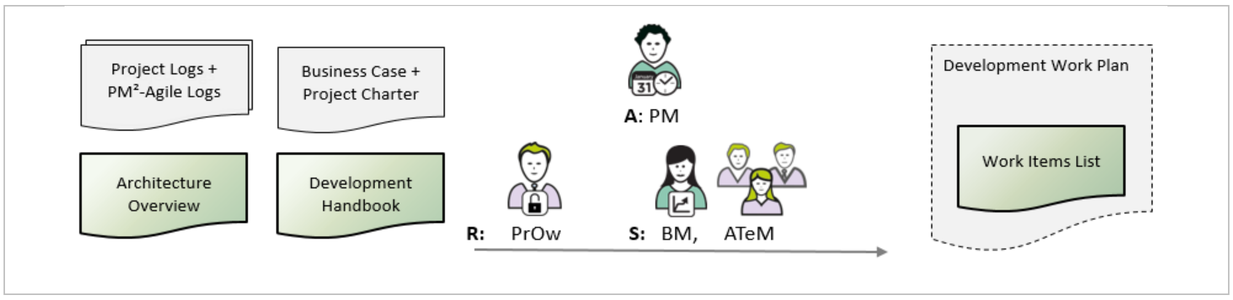


**Fig. 7.9** Work Items List — Inputs and main roles

| PM² Lifecycle Phases: | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Work Items List | - | *Created* | *Updated* | - |

### *7.2.2.2 Release Plan*

A Release Plan documents the high-level UX and development milestones. It gives information of what will be released and when, based on the information the Project Core Team (PCT) has in a specific point in time. The Release Plan results from the initial overall project planning effort during the PM² Initiating and Planning phases and should be adjusted according to the progress and feedback from the Agile Project Core Team (A-PCT).

The main purpose of the Release Plan is to:

- plan the release of usable versions of the solution;
- provide a high-level plan with the main goals to be achieved, including major features to deliver;
- offer visibility to the risks, issues, decisions, changes, constraints and assumptions related to the development aspect, such as dependencies on other projects and other information systems;
- assist when scheduling transition activities (e.g. deployment strategy and activities);
- assist when scheduling Business Implementation activities (e.g. training activities).

Releases introduce a timeboxed rhythm to the construction efforts, providing (some level of) predictability for both the Agile Project Core Team (A-PCT) and the rest of the stakeholders.

The Release Plan facilitates the management of dependencies that the information system being developed may have with other projects, information systems or operations activities.

For each planned release, transition activities must be defined, the environment targeted by the release is prepared and the end-users are ready to use the delivered solution. These activities should be documented and aligned with the PM² Transition Plan.

Releases that are meant to be made available to the overall requestor organisation must also have associated the needed Business Implementation activities, ensuring a successful information system's roll-out. These activities should be documented and aligned with the PM² Business Implementation Plan.

| Key Participants | Description |
|---|---|
| Project Manager (PM) | Accountable for the alignment between the delivery strategy reflected in the Release Plan and the overall project vision and all relevant milestones described in the Project Charter. |
| Team Coordinator (TeCo) | Responsible for the development of the Release plan, supported by the PrOw - who provides guidance based on the prioritised Work Items List (WIL) - and the remaining Agile Team Members (ATeM), who provide additional inputs on viability of the plan. |
| Agile Team Member (ATeM) | Supports the Team Coordinator (TeCo) by providing inputs related to the plan's viability, including dependencies between the several Work Items. |
| Architecture Owner (ArOw) | Supports the Team Coordinator (TeCo) by providing inputs related to the solution's global architecture and building blocks that may exist and impact the Release Plan. |

| | |
|---|---|
| Business Manager (BM) | Informed by the Product Owner (PrOw) on the updated Release Plan. |
| Product Owner (PrOw) | Supports the Team Coordinator (TeCo) by providing the delivery strategy of the business as a set of milestone dates, the release objectives and foreseen features and the corresponding Work Items. |

**Guidelines**

- **Build a Release Plan as early as possible** – As the main driver to deliver the solution, hold the first Release Planning ceremony to establish the initial Release Plan as soon as possible, preferably in the beginning of the Planning phase.
- **The Release Plan reflects the Solution's release strategy** – The Release Plan is more than just a calendar of dates for major releases with the key functionalities. With the Release Planning ceremony playing a critical role, the Release Plan reflects the overall strategy (marketing, operational, economic, legal, etc) to release the solution to the users' community.
- **Provide different levels of detail** – Because different types of audience are interested in the Release Plan, different levels of detail must be provided. For example, the Appropriate Governance Body (AGB) is interested in the several releases planned and their major objectives to validate if the Release Plan is aligned with the overall strategy. However, the Product Owner (PrOw) needs to know which work items will be delivered in each release, as this will help to verify which part of each feature will be released and when.
- **Revisit the plan after each Iteration** – At the end of each iteration, some new inputs will generate new work items in the Work Items List (WIL) as part of the learning process materialised by ceremonies like the Iteration Review and Iteration Retrospective. As an example, the updated data regarding the velocity of the team must be reflected in the Release Plan in order to continuously evaluate its feasibility.
- **Release Planning ceremony** – The outputs of the several steps of the Release Planning ceremony contribute to an updated Release Plan. Work items added, updated or removed from the Work Items List (WIL) with the corresponding relative estimates provided by the Agile Team Members (ATeM) must be incorporated in the Release Plan to reflect the most up to date scenario;
- **Release Plan Feasibility** – New facts, data, decisions, will constantly surface throughout the project impacting the size of the Work Items List (WIL) and the team's velocity. These two elements evaluate the Release Plan's feasibility and allow the business stakeholders to make adjustments when required. Also, using different sets of data can help creating different possible scenarios, like using an optimistic and pessimistic velocity.
- **Have a Deployment Plan** - Plan the strategy for deploying the software (and its updates) into the production environment as early as possible, because it may have an impact on the Work Items List (WIL) and the Release Plan. This is also true when specific operations and support departments are responsible for managing the overall corporate deployment system. Having a Continuous Delivery Pipeline as part of a DevSecOps initiative can streamline this process and dramatically reduce its complexity.
- **Make assumptions clear and visible** – The velocity of the team and the pre-defined size of each iteration are two examples of elements that impact the way a Release Plan is understood. When defining the Release Plan, these elements must be clearly visible and identified as the current Release Plan's assumptions.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Release Plan | I | **A** | **R** | **S** | **S** | **S** |

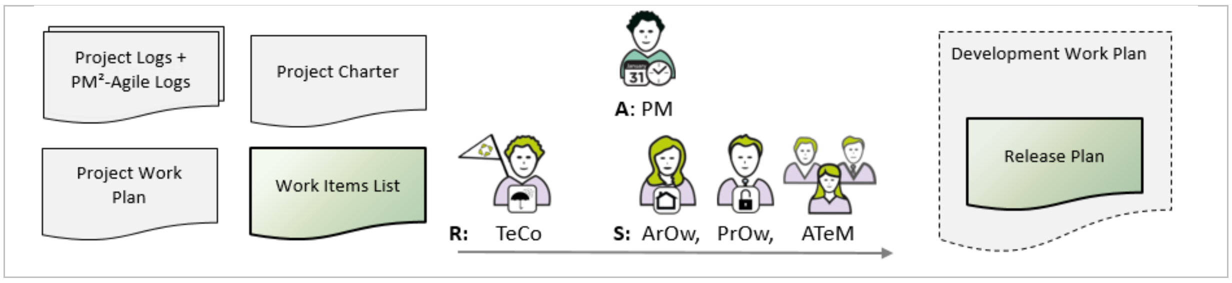


**Fig. 7.10** Release Plan — Inputs and main roles

| PM² Lifecycle Phases: | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Release Plan | - | *Created* | *Updated* | - |

#### *7.2.2.3 Iteration Plan*

In its minimal form, the Iteration Plan is the set of Work Items - a subset of the Work Items List (WIL) - which are planned for the upcoming iteration. It includes the Iteration start and end dates, the Iteration Goal, defined by the Product Owner (PrOw) and committed to by the Agile Project Core Team (A-PCT) and the list of all the work items that must be delivered to achieve the Iteration Goal. Additionally, the Iteration Plan may need to capture synchronisation points with other teams, especially if they are in different projects. This artefact is also used to refer to issues, risks, etc, that need to be solved during the iteration.

The Iteration Plan helps the Agile Project Core Team (A-PCT) monitoring the progress of the iteration (using an Iteration Kanban board and burndown charts, for example), and keeps the results of the iteration for assessment that may be useful for future improvement, based on the outputs of the Iteration Review.

The main purpose of the Iteration Plan is to provide the Agile Project Core Team (A-PCT) with the following:

- One central source for information regarding the iteration goal.
- A visualisation of the scope of an iteration.
- Evaluation results.
- An indicator of the current team's progress and a forecast for the iteration work completion.

Work items assigned to an iteration do not necessarily have the same priority. Once Work Items have been assigned to the iteration, the team ensures that they can complete all work, regardless of original work item priorities (except for items that represent blocking dependencies with items from other teams). Deciding what to develop first in an iteration will vary across projects and iterations.

Work Items can be part of a feature (functional unit) that should represent a recognisable and valuable part of the solution. The Work items are then decomposed in the corresponding tasks required to complete them.

| Key Participants | Description |
|---|---|
| Project Manager (PM) | Accountable for ensuring the Iteration Plan goal is aligned with the Release Plan strategy and the Project Charter. |
| Team Coordinator (TeCo) | Responsible for the development of the Iteration plan, assisted by the Product Owner (PrOw), providing the Iteration goal and the remaining Agile Team Members (ATeM), providing information on what items will be implemented and how. |
| Agile Team Member (ATeM) | Supports the Team Coordinator (TeCo) by providing inputs related to the feasibility of the plan, including which items can be implemented during the iteration. |
| Architecture Owner (ArOw) | Supports the Team Coordinator (TeCo) by providing relevant information regarding potential architecture risks/issues that may impact the iteration plan. |
| Product Owner (PrOw) | Consulted by the Team Coordinator (TeCo) to provide the iteration goal and a prioritised Work Items List (WIL) - the main input for the iteration plan. |
| Business Manager (BM) | Informed by the Product Owner (PrOw) and Project Manager (PM) on the iteration goals and what can be expected by the end of the Iteration. |

**Guidelines**

- **One Iteration, one (Iteration) Plan** – Every Iteration starts with the Iteration Planning ceremony, where a strategy to reach an iteration goal is defined, and ends with the Iteration Review ceremony, where the delivered work’s result is evaluated towards the defined iteration goal. This information is captured in the corresponding and unique Iteration Plan.
- **Start with the Iteration Planning ceremony** – The strategy defined by Agile Project Core Team (A-PCT) during the Iteration Planning ceremony provides critical information to the Iteration Plan,

such as the iteration period, the iteration goal, the work items to implement, dependencies, associated risks, etc. The initial version of the Iteration burndown chart[18] is also generated based on the inputs coming from this critical ceremony.

- **It's a living artefact** – The Iteration Plan is not a static artefact since some of its elements will be frequently updated. For example, the burndown chart must constantly reflect the iteration's situation based on the planned and achieved work. The iteration's scope is another example of an element that may be subject to revision, generating several changes in the Iteration Plan.
- **Close with the Iteration Review** – The Iteration Review ceremony presents final conclusions about the work done by the Agile Project Core Team (A-PCT) during the iteration. Was the Iteration Goal achieved? Were there any work items not delivered successfully? If so, what was the reason? Collected during this ceremony, this invaluable information is recorded as part of the Iteration Plan for present and future reference.
- **Use of software tools to create and maintain an Iteration Plan** – Before compiling the different types of information described in the previous points to a standard, manually generated document, one should investigate alternative software tools that may save a significant amount of time.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Iteration Plan | I | **A** | **R** | C | **S** | **S** |

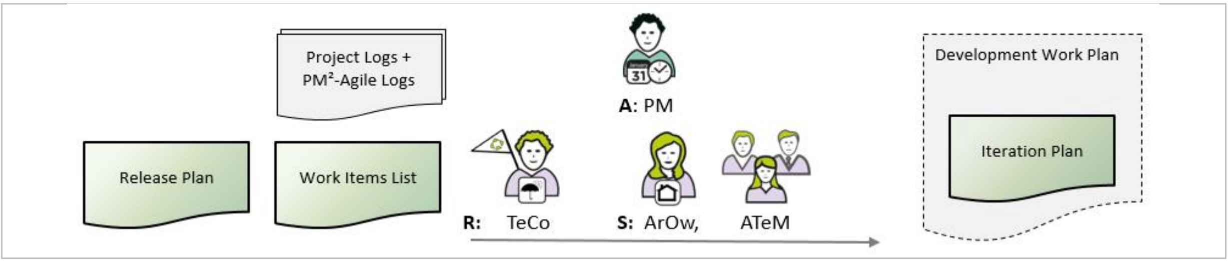


**Fig. 7.11** Iteration Plan — Inputs and main roles

| **PM² Lifecycle Phases:** | **Initiating** | **Planning** | **Executing** | **Closing** |
|---|---|---|---|---|
| Iteration Plan | *Created* *Updated* | *Created* *Updated* | *Created* *Updated* | - |

## 7.2.3 Test Plan

A Test Plan defines the goals/scope/objectives of tests, the targeted items, the approach to be taken, the resources required, and the deliverables to be produced. It specifies the universe of tests to be performed (e.g. unit tests, regression tests, functional tests, integration tests, user acceptance tests) as well as the stages in which tests will be executed (e.g. development, integration, test, acceptance). Testing techniques, metrics and criteria used to assess the test results and test completion are also detailed.

The main purpose of having a Test Plan is to outline and communicate the testing efforts for a given schedule.

Every Test Plan forms the framework within which the Agile Team Members (ATeM) performing the testing will work for the given schedule. By directing, guiding and constraining the test effort, the plan focuses on the necessary deliverables and communicates the effort's intent to stakeholders.

Therefore, a Test Plan should avoid details that would not be understood, or would be considered irrelevant by the people involved in the testing activities. Because different levels of tests focus on several areas of functionality and risk, other organisational units may possess various responsibilities regarding the overall testing efforts. Detailed test plans are then necessary for distinct organisational units to plan, prepare, execute, and report testing without increasing risk.

[18] For more information on the Burndown chart, please check the "Burndown Charts" tool/technique.

| Key Participants | Description |
|---|---|
| Team Coordinator (TeCo) | Supports the drafting of the Test Plan. Ensures alignment with the project-level Project Handbook, in close collaboration with the Project Manager (PM). |
| Agile Team Member (ATeM) | Responsible for drafting the Test Plan, while providing technical advice regarding the testing activities, the scope and type of testing, testing techniques, etc. |
| Architecture Owner (ArOw) | Supports by providing technical advice and identifying technical constraints that may impact the testing activities. |
| Product Owner (PrOw) | Supports providing insight and information related to the requestor organisation's reality that may impact the testing activities. |
| Project Manager (PM) | Accountable for how the Agile Project Core Team (A-PCT) plans to address the Testing aspect of the project. Also ensures the alignment of the testing efforts with the overall management of the project deliverables' acceptance. |

**Guidelines**

- **Define a Global testing strategy** - Generate an outline of a Master Test Plan early in the project. Testing efforts should not be seen as extra-work to be done but rather incorporated within the development cycles (iterations, releases) as routine work.
- **It's a team effort** – Just because it's a Test Plan, it doesn't mean it's the " Testers' responsibility". Discuss the test mission with the entire Agile Project Core Team (A-PCT). After gaining initial agreement on the test mission, goals, and specific objectives, define the test approach by outlining this information in the Test Plan.
- **Align with the Development Handbook** –When defining the global testing strategy, make sure that it's compatible with the development strategy described in the Development Handbook.
- **Focus on the communication aspect of the testing efforts** - Testing may include many different stakeholders at different moments and it is important that everyone understands the testing approach, namely the expected involvement of the stakeholders during the testing efforts.
- **Find the right balance** - Review the Test Plan and all related elements with all stakeholders to obtain agreement. Focus on finding a balance between the testing resources and the scope of the current testing effort, based on the identified risks and priorities.
- **Document testing risks and contingencies for risk mitigation in the Test Plan** - If project specific risk planning is part of a more significant risk management effort, coordinate testing risks with the risk analysis, design, and mitigation processes of the Project Manager. Ensure that test-related risk items are accurately traced back to requirements and testing assets.

| **RAM** (RASCI) | BM | PM | TeCo | PrOw | ArOw | ATeM |
|---|---|---|---|---|---|---|
| Test Plan | I | **A** | **S** | **S** | **S** | **R** |

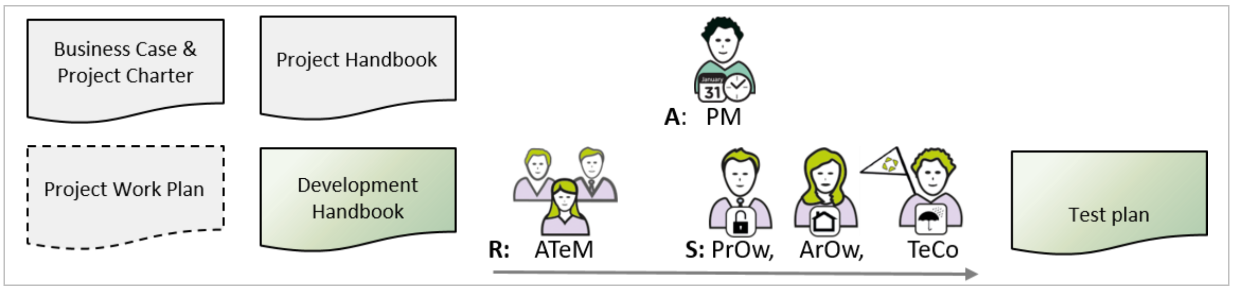


**Fig. 7.12** Test Plan — Inputs and main roles

| PM² Lifecycle Phases: | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Test Plan | - | *Created* | *Updated* | - |

### 7.2.4 Deployment Plan

The Deployment Plan defines the strategy, the roles and responsibilities, and the tasks that must be taken into account to ensure the information system can be deployed, run and managed by the IT operations team. Deployment efforts can impose a great deal of change and stress, both in the IT operations environment and on the information system's end users, especially if not part of a well-defined and structured deployment strategy.

Because the Deployment Plan focuses on the deployment effort's technical aspects, it is closely related to IT operations such as cut-over, rollbacks, back-out procedures for contingency and risk management.

The PM² Transition Plan and Business Implementation Plan complement the Deployment Plan by covering the strategy to ensure that the client organisation is ready to use the system (training, support, etc.)

The primary purpose of the Deployment Plan is to:

- outline and communicate the strategy applied for the deployment efforts during the project life cycle;
- create consensus and awareness of the efforts and responsibilities involved in a successful technical deployment of the solution – IT operations.

The Deployment Plan should consider the iterative and incremental nature of PM²-Agile. Furthermore, deployment efforts should contribute to the overall goal of releasing working solutions to the information system's users at regular intervals. The Deployment Plan must have enough detail to allow the IT operations team to deploy, run, and manage the information system.

The Deployment Plan is closely related to the Operational Model as both artefacts are vital elements to ensure that the information system can be deployed and used in production environments The Deployment Plan should include the following sections:

- **Deployment in Production** – Details the complete deployment process, including which features are expected to become available, which work packages will be generated, which libraries will be used, environment requirements, generated configuration data, etc.
- **Validate the Solution** – Describes the used mechanisms to guarantee that the deployment was done correctly. Specific stress and performance tests, smoke tests with some key features can verify if a correction is required, possibly resulting in a full roll-back of the deployment.
- **Monitor, Report and Respond to problems** – Describes what are the foreseen mechanisms to help detecting, reporting and solving several different problems. The goal is to identify potential issues *before* they become *real* problems and jeopardize the correct system deployment and operation.

| **Key Participants** | **Description** |
|---|---|
| Team Coordinator (TeCo) | Responsible for creating the Deployment Plan aligned with the Development Work Plan (Release Schedule) and with the project-level Transition Plan, in close collaboration with the Project Manager (PM). |
| Agile Team Member (ATeM) | Supports by providing technical advice related to the deployment efforts (data migration, contingency actions, or continuous delivery pipeline details). |
| Architecture Owner (ArOw) | Supports by providing technical advice and identifying technical constraints that may impact the deployment approach to be defined. |
| Product Owner (PrOw) | Supports by providing insight and information related to the requestor's organisation reality that may impact the deployment approach to be defined. |
| Project Manager (PM) | Accountable for the Agile Project Core Team's (A-PCT) deployment approach and the alignment of the deployment efforts with the overall transition activities. |
| Representative of the relevant operational unit | Contributes by bringing information related to operational processes and infrastructure, checking the feasibility, effectiveness and efficiency of the Deployment Plan and associated risks. |

**Guidelines**

- **Compatible with the Development Handbook** – If the Development Handbook foresees the incremental and iterative development nature of PM²-Agile, the Deployment Plan must foresee a clear strategy to enable this approach. The setup of a Continuous Delivery Pipeline is a strategic decision that must be described and validated in the plan.
- **Development and Operations** – The DevSecOps mindset translates to a set of practices involving Software development and Operations. It brings these ecosystems together to create the Deployment Plan. However, if the organisation is siloed with an independent IT Operations department, separated from Development teams, then IT Operations should be integrally involved to account for operational constraints.
- **It's a living artefact** - Revisit the Deployment Plan as the project evolves and feedback from previous deployment efforts is gathered.
- **Align with the Release Plan** - Ensure that the deployment efforts are planned with respect to the defined Release Plan. If a specific IT Operations department manages deployments in the production environment with no DevSecOps practices in place, this alignment becomes crucial.
- **Test it first** – Assure the Operations team can understand the Deployment Plan and successfully deploy, operate and manage the solution in a pre-production environment. Perform several *dry-runs* to guarantee a seamlessly transition to Production.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Deployment Plan | I | **A** | **R** | **S** | **S** | **S** |

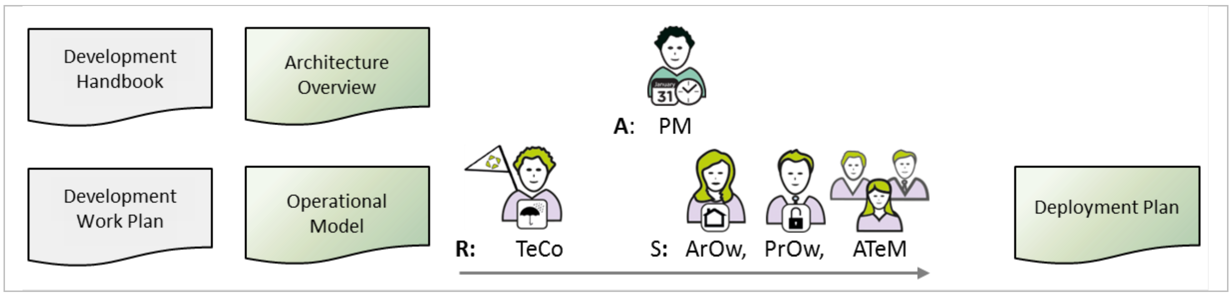


**Fig. 7.13** Deployment Plan — Inputs and main roles

| **PM² Lifecycle Phases:** | **Initiating** | **Planning** | **Executing** | **Closing** |
|---|---|---|---|---|
| Deployment Plan | - | *Created* | *Updated* | - |

## 7.3 Coordination & Reporting Artefacts

### 7.3.1 Development Status Report

The Development Status Report documents and communicates the status of the Agile Project Core Team (A-PCT) towards the development goals and it's incorporated at the project reporting level.

Similar to the Project Status Report, the Development Status Report includes tracking information regarding the cost, schedule, scope, risks, issues, changes, and forecasts for the project's next steps. The Development Status Report can follow each iteration and/or each release, depending on the reporting needs defined for the project.

One of the key elements of the report is the burndown chart, both at the iteration and release level. The release burndown chart shows the estimated functionality remaining to complete the current release and It provides answers to the following questions:

- When can a release be completed based on the team's performance (velocity)?
- What progress has been accomplished so far?
- Is the team's velocity sufficient to complete the release on time?

The **release burndown chart**[19] provides a broad view of a release's progress. It is reviewd, at least, once each iteration, and indicates if the team is delivering functionality at the expected pace. It can also expose projects whose scope is out of control and thus, negatively affecting the expected progress. Based on a trend or forecast, the chart allows the team can take action to better control scope, or adjust resources, budget, timelines, quality levels or commitments.

The **iteration burndown chart** is the primary tool for understanding the status of the current iteration. It shows the current team's performance, based on estimated effort left versus the optimal team's performance, based on the expected effort left that guarantees the iteration goal's fulfilment.

The iteration burndown chart should be updated frequently, preferably daily. Daily or frequent updates allow the team to react to changes. For example, changes can include reducing the project scope by removing Work Items from the iteration or the Work Items List, reducing the ambition level associated with a work item, or finding better ways of approaching Work Items.

Unlike the iteration burndown chart that tracks remaining effort in points or hours, the release burndown chart tracks functionality to determine whether the team is delivering working software in each iteration. The resulting release burndown chart helps comparing how much functionality the team is delivering with how much they forecasted to deliver.

| **Key Participants** | **Description** |
|---|---|
| Team Coordinator (TeCo) | Responsible for preparing the Development Status Report and ensuring the alignment of the development-related progress information with the project-level Project Status Report, in close collaboration with the Project Manager (PM). |
| Agile Team Member (ATeM) | Supports the preparation of the Development Status Report by providing information on the work they perform daily. |
| Project Manager (PM) | Accountable for the Development Status Report, the Project Manager (PM) must be aware on how the Agile Project Core Team (A-PCT) is progressing. She also ensures that the Project Status Report's relevant information is captured from the Development Status Report. |
| Product Owner (PrOw) | Is informed about the progress of the team. |

**Guidelines**

- **Refer to Development Work Plan** – The Iteration and Release Plans provide critical and transparent data for a precise and objective Development Status Report. Team's velocity, burndown charts, Work Items List and an updated Release strategy are some of the elements available.
- **Project Management Logs are also vital** – Issues, decisions, risks are all part of a project's day-to-day life and must be managed closely. Keeping the project management logs current ensures everything is addressed and resolved.
- **Align Progress meetings with Iteration and release cycles** – Because every iteration provides objective information on the Agile Project Core Team' (A-PCT) actual progress, having a progress status meeting at the end of each iteration is an effective way to communicate the development effort's status.
- **Transparency** – Always keep the status information visible to stakeholders and the project team in a project workspace (walls or automated tool), where stakeholders can come and experience first-hand the progress being made by the team.

| **RAM** (RASCI) | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Development Status Report | I | **A** | **R** | I | I | **S** |

[19] For more information on Burndown chart, please check the "Burndown charts" tool/technique

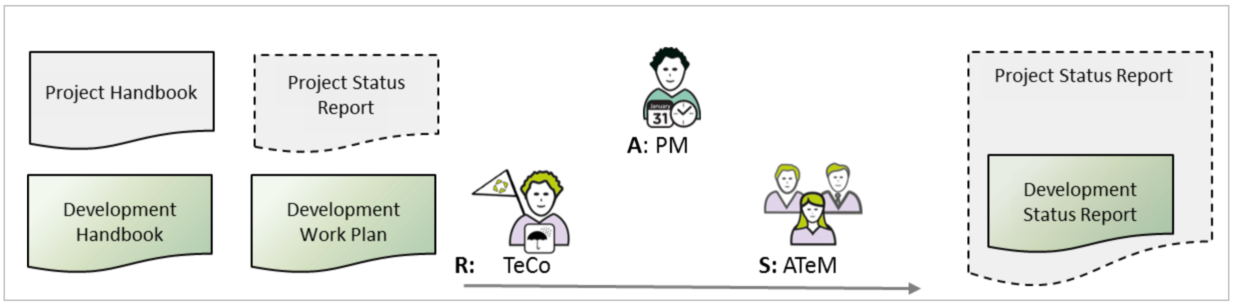


**Fig. 7.14** Development Status Report — Inputs and main roles

| **PM² Lifecycle Phases:** | **Initiating** | **Planning** | **Executing** | **Closing** |
|---|---|---|---|---|
| Development Status Report | - | *Created* | *Created* | - |

### 7.3.2 Project Logs

To enable the overall monitoring of the project' performance during its executing phase, the Agile Project Core Team (A-PCT) should also document and manage domain-specific risks, issues, decisions and changes. While managing this information, the Project Core Team (PCT) can identify items that may impact the overall project performance and need to be addressed within the general project organisation.

When managing these logs, the Project Core Team (PCT) can support the Project Manager (PM) to report and forecast project evolution to the project stakeholders. The project logs are also the coordination tools used by the Project Manager to assign items to the Agile Project Core Team (A-PCT) to be further evaluated and managed.

As a starting point, the PM² related Management Plans and the PM² associated artefacts can be used to ensure that information regarding domain-specific risks, issues, decisions, and changes are properly managed.

The table below indicates the PM² management plan and respective logs for each item.

| | **Reference** |
|---|---|
| Risks | Risk Management Plan, Risk Log |
| Issues | Issues Management Plan, Issue Log |
| Decisions | Decision Log |
| Changes | Project Change Management Plan, Change Log and Change Request Form |
| Testing Log | Test Plans and Deliverables Acceptance Plan |
| Retrospectives Log | Project-End Report |

### 7.3.3 Project Reports

The development aspect of the project should contribute with domain-specific development status and progress information to the Project's reports. The goal is to translate the domain-specific information and metrics used to monitor the development progress and incorporate it at the project level status and project reports.

The Project Status Report introduces the feedback of the development aspect of the project to the Managing Layer in the section related to the Development Status Report.

For projects that follow PM²-Agile, the Project Progress Report is a frequent reporting tool (e.g. bi-monthly or quarterly as opposed to yearly for multi-annual projects) that allows the Project Core Team (PCT) and the overall project to better capture and manage significant scope changes.

As in PM², the Project Progress Report provides a high-level overview of the entire project and its actual status.

The report includes a Project Overview (Project stakeholders, Milestones and Deliverables, Project Plan, Budget and Costs) and further Project Details (Scope Changes, Major Risks/Issues and Actions Taken, Achievements).

Providing this high-level overview of the entire project, including domain-specific considerations, together with updates to the initial Project Charter (PC) allows communicating to all the stakeholders how the project is progressing, what has evolved since the initial charter and the reasons that lead to those changes. Ad hoc Project Status reporting (including an updated version of the Project Charter) may be needed, depending on the complexity of the project and the impact of scope changes during the development of the solution.

# 8 Tools & Techniques

This section describes, from an introductory, high-level perspective, some of the most widely used tools and techniques in PM²-Agile to improve quality and enhance product development agility.

A more detailed description of these tools and techniques can be found in their specific areas of knowledge and communities of practice.

## 8.1 Assessing Teamwork

To be truly effective, a team must overcome several obstacles, as outlined by Patrick Lencioni in his best-selling The Five Dysfunctions of a Team. Patrick describes this process as a set of steps, built on top of each other, that allow a group of people to become an effective, mature and high performing team.

This assessment provides leaders with an opportunity to explore and overcome the pitfalls that side-track teams. It gives the team members a sense of their unique strengths and identifies areas for improvement in each of the five key fundamentals of developing a cohesive and productive team: trust, conflict, commitment, accountability, and results.

## 8.2 Self-Organising Teams

Self-organizing teams have the ability and authority to take decisions and readily adapt to changing demands. From a social systems perspective, this means that the team can create new approaches and adapt to meet new challenges in their environment. Self-Organising is a process and not something that is done once and for all.

Promoting self-organising teams within an organisation is a strategic decision. Managers must create the conditions that enable teams to thrive and continue to self-organize. They must commit to guide evolving behaviours that emerge from the interaction of independent agents instead of specifying in advance what effective behaviour is.

It takes a combined commitment and effort from many different levels of the organisation (Senior Management, Project Management, and Development/Team Management) to help teams emerge as self-organising and high performing.

## 8.3 Design blocks

Design blocks are part of an iteration as a starting point to challenge the requirements that are seen as assumptions for each feature. Design blocks are based on design thinking techniques, focused on materialising the work items that need to be delivered in the next iteration.

From a practical point of view, Design Blocks take the form of a set of workshops where the Agile Project Core Team (A- PCT) and other stakeholders meet, discuss and declare the assumptions related to a feature or the work items for the next iteration.

The facilitator and coordinator (UX specialist) drives all the participants to prepare a low fidelity prototype, by sketching the future feature altogether. The UX specialist makes sure that the design, interaction, behavior, requirements and technical matters are thoroughly discussed, understood and agreed upon by everyone.

When the Agile Project Core Team (A-PCT) feels confident on the feature's specific low-fidelity version, the User Interface (UI) designers provide a high-fidelity prototype, where UI standards from existing design libraries are applied. This high-fidelity prototype will then drive the coding of the feature, as part of the daily work of the team.

## 8.4 Features and Stories

Features and Stories are a way to express the requirements that a given solution should address. In general, they capture the description of new functionality (or updates to existing functionality) by starting at a high level and get broken down into smaller items as more detail is needed and understood.

Features represent services/functionalities delivered by a solution to tackle the stakeholders' needs, while Stories are small pieces, corresponding to vertical slices of system's desired functionality. Implementing a feature is a more complex and lengthy process than implementing a story. This is because a feature includes all relevant scenarios, making it very unlikely to be implemented within an iteration's scope.

Since the Agile Project Core Team (A-PCT) uses iterative development as a way to help ensuring effective incremental development, it becomes a mandatory exercise to decompose every feature in the corresponding stories so that they can be integrated in the Work Items List (WIL).

Stories are very convenient because they were thought in a way that allows them to be implemented in a short period while delivering value to the stakeholders. Stories are the ideal tool for Agile teams because it allows them to build the solution “piece by piece” while collecting, analysing and integrating feedback in short iterations, which is a great way to validate if the team is progressing in the right direction.

Another key benefit of the stories is that the simplicity of thought and structure allows a team and other stakeholders to manage the project’s scope efficiently. Stories make it easy to understand what has been implemented and delivered and plan what is left to be done.

## 8.5 User Stories Breakdown

User stories may be of different sizes during a project lifecycle. When a user story is about to be implemented, it should be right-sized to facilitate estimation, prioritisation and development within a reasonable time-frame (an iteration). But this is just one side of it. There are several other benefits that make this tool a very important one to master.

- **Modular Implementation** – Working with smaller user stories helps improving design. Improved design contributes to higher levels of maintainability and promotes easier scalability.
- **Easier response to change** – Smaller user stories means smaller changes, with less impact and easier to implement. Therefore, easier to incorporate.
- **Risk Reduction** – Because smaller stories are simpler and easier to describe and demonstrate, they help reducing one of the biggest risks in a project: uncertainty.
- **Reduced Variability** – Because smaller stories are simpler and have less variability, they often do not delay delivery and travel faster through the implementation process (the *System*), reducing queues’ length.

## 8.6 Definition of Done

In an iterative and incremental development environment, the development team has to agree on the conditions that must be satisfied for a given Work Item to be considered complete. An agreed-upon "Definition of Done" (DoD) consists on a set of items that the Agile Team Members (ATeMs) check to determine if an implemented Work Item complies with the team's quality standards.

The DoD should result from the development team’s discussion and agreement of what "done" means in the context of a project. In some organisations, a common "definition of done" may be available and can be applied by different development teams, moving towards cross-project understanding of what "done" means for the organisation.

Any Agile Project Core Team (A-PCT) benefits from having a DoD for their Work Items because it also brings the notion of quality outputs into the development cycle and reduces the likelihood of having problems (e.g. defects) in a later stage of the information system lifecycle.

## 8.7 Planning Poker

Planning Poker is a consensus-based technique for relatively estimating the size of the development goals. It is an efficient way of reaching agreement, without spending too much time on a specific topic, enabling team members to express their opinions, thoughts and concerns.

This technique should ideally be used in a workshop environment to refine and estimate higher priority work items.

## 8.8 Kanban Method

Kanban is a technique used in Lean manufacturing. It means "Board" in Japanese and it’s an essential tool used in visual management. It was first used in Toyota’s Production System for cars, enabling the upstream operations to produce a part only when it was required by the downstream operations. This reduced the work in progress inventory and improved the process efficiency.

Kanban provides a visual representation of the process using a board and cards, explaining what and when to produce or add a feature during the build process. In traditional push methods, inventory waste is built-up during the production process as bottleneck are experienced in the production chain. In the pull method, the last step in the production chain “tells” the previous step when it is ready to produce, and receives the work finished in the previous step. This pull is then passed back, step-by-step, to the first step.

There are several benefits of using the pull method for development and using Kanban to control and visualise the production chain. Substantially fewer errors, less time to completion, decreased risk, increased visibility and transparency, and less inventory in waiting are just a few of the advantages.

However, using these methods requires strict organisation and a diligent Agile Project Team (A-PCT).

## 8.9 Burndown Charts

Burndown charts are visual tools used to track different dimensions of progress in an Agile project. These charts show how a planned and measurable item (e.g. effort hours or story points) is being ‘burnt down’ to zero (no work left).

- Burndown charts track the progress of iterations (effort hours or story points across days) and the progress of releases (story points across iterations), highlighting any obstacles that might hinder the team’s performance.
- Burndown charts are dynamic, in the sense that the planned work may change as a result of the feedback received during development.
- Burndown charts should be highly visible, enabling Project Core Team (PCT) members and the broader project organisation to see the current progress and forecast with complete transparency.

## 8.10 Test-Driven Development

Test driven development (TDD) is the practice of writing developer tests and implementation code, concurrently, in fine level granularity.

In Test-Driven Development, the developer first writes a small test to validate a piece of code and then runs the test to confirm that it fails (a sanity check). After, he writes just enough implementation code to make that test pass successfully. This cycle is short and it rarely goes beyond 10 minutes.

In each cycle, the tests come first. Once a test is done, the developer goes on to the next test until there are no more tests to be written for the implementation of the work item currently under development. This also ensures that developers only write the necessary code to implement each work item, as they will be focused on creating the code that validates the tests created.

Another significant advantage of TDD is that it enables taking small steps when writing software, which is safer and far more productive than writing code in large increments.

## 8.11 Pairing

Pairing is a technique that implies two Agile Team Members (ATeMs) working together, collaboratively, on a particular activity, such as designing a subset of the solution, coding or testing a piece of functionality, etc. When the activity subject to pairing is about programming a piece of the system (e.g. a particular user story), the technique is known as Pair Programming. In this case, both developers focus on the code being produced.

Pair programming does not assume a coaching or training relationship as both developers are peers working on the same project. Naturally, if a pair includes a senior and a junior developer, some degree of on-the-job training will take place as the more experienced developer supports the junior developer producing quality software.

Pairing (and Pair Programming in particular) provides Instant and real-time work review. It brings higher quality to the deliverables as two peers discuss and explore how to best address what needs to be implemented.

This page has intentionally been left blank

# Appendix A: Acknowledgements

The European Commission is grateful to all those who have contributed to the development of the current version of the PM²-Agile guide.

**Produced by:**

European Commission

DIGIT.B4 Software Engineering Capabilities

| | |
|---|---|
| VEKEMANS Tom | Head of Unit |
| VIJGHEN Philippe | Deputy Head of Unit |

DIGIT.B4.001 Project Sourcing and Shaping Sector

| | |
|---|---|
| VAN GAEVER Alain | Head of Sector |

*Centre of Excellence in PM² (CoEPM²)*

| | |
|---|---|
| BERGHMANS Marc | PM² Ambassador |
| MICHELIOUDAKIS Elias | Senior Consultant |
| PALHOTO Tiago | Senior Consultant |
| LOPES António | Senior Consultant |
| WHYE Gregory | Senior Consultant |
| MODI Nina | Senior Consultant |
| COOPER Veronica | Communications Consultant |
| MICHOTTE Alexandra | Design and Graphics |

This page has intentionally been left blank

# Appendix B: Additional Resources

## B.1: The PM²-Agile Principles

**1. The highest priority is to satisfy the client through early and continuous delivery of valuable solutions.**

The goal of solution delivery should be the delivery of an actual consumable and valuable solution. Agile teams move away from a strategy where they try to think through all the details upfront, thereby increasing both project risk and cost. Instead, they invest a bit of time thinking through the critical issues, allowing the details to evolve over time as they learn through incremental creation of the solution.

**2. Changing Requirements are welcome**, **even late in the solution delivery lifecycle**.

Agile processes harness change for the stakeholders' advantage. Requirements will change throughout a project. Traditional project teams often adopt change management processes designed to prevent/limit scope creep. However, these are in fact change prevention and not change management processes. An Agile change management approach ensures that functionality is developed following a well-established priority and requirements evolve to reflect stakeholders' improved understanding of what they need.

**3. Deliver value frequently through working solutions.**

Agile teams should deliver value at the end of each iteration. Iterations' length should be between two and four weeks, with a preference for the shorter.

Frequent delivery of a workable solution gives business stakeholders the opportunity to contribute with timely feedback, keeping the project transparent and allowing adjustments in the project's direction when needed.

**4. Business people and project team must work together throughout the project.**

Regular access to the project stakeholders or their representatives is crucial for a successful project. Agile teams adopt practices such as on-site client and active stakeholder participation, and adopt tools and techniques that enable stakeholders to be actively involved in solution delivery.

**5. Create teams with motivated individuals. Give them the environment and support they need to self-organise, and trust them to get the job done.**

Too many organisations have a vision that they can hire hordes of relatively unskilled people, provide them with a CMMI or ISO-compliant process description, and they will successfully develop solutions. This does not seem to work that well in practice. Agile teams, on the other hand, realise that a team has to be built with people who are willing to work together collaboratively and learn from each other. They have the humility to respect each other and realise that people are a primary success factor in solution delivery. They should be allowed to create an environment which fosters collaboration, use of tools that they find most effective, and have the freedom to customise and optimise their team's development process.

**6. The most efficient and effective method of communication is face-to-face conversation.**

For a delivery team to succeed its members must communicate and collaborate effectively. People can communicate in many different ways, and face-to-face communication in a shared and collocation based environment is often the most effective way to do so. Endless emails and exhaustive documents are far less effective than the immediate feedback of conversation.

Distributed teams cannot be an excuse for reverting back to extensive documentation practices since video chat or other communication systems can be used to support face-to-face conversations.

**7. The primary measure of progress is the usefulness of what has been delivered.**

The delivery of a useful, consumable solution that provides value to the project stakeholders is the primary measure of project progress. This solution should meet their changing needs and not some form of 'earned value' measure based on documentation's delivery or the holding of meetings.

**8. Continuous attention on quality.**

Following the principle of continuous attention to quality, Agile reduces corrective measures such as test by adopting practices so that quality is built-in, reducing the need for fixing defects. Agile practitioners know that they need to start with high-quality work products and keep the quality high via refactoring, full regression test suites, etc.

**9. Simplicity – the art of maximising the amount of work not done — is essential.**

Agile developers focus on high value activities and strive to maximise their stakeholders' return on investment. From a Lean point of view, simplicity is essential so that only the most important things are worked on, reducing variability and the associated delays.

**10. At regular intervals, the team reflects on how to improve, then tunes and adjusts its behaviour accordingly.**

Agile teams reflect frequently (at least by the end of each iteration) on how they are performing as a team. They adopt techniques such as retrospectives to reflect on their practices, generate insights about how they can improve and define concrete actions to materialise those insights, preferably sooner rather than later.

When such a short feedback loop is an inherent part of the process through an established ritual, the sensitivity to inefficient and ineffective patterns is higher, and lessons learned are acted upon on a regular basis, not postponed indefinitely.

**11. Agile processes promote sustainable development. The Project Team should be able to maintain a constant pace indefinitely.**

Just as people cannot sprint for an entire marathon, a solution cannot be developed by forcing people to continuously work over their capacity.

**12. Agile practices should be enterprise-aware, taking into consideration IT governance, enterprise architecture, and interoperability requirements. Agile teams should be able to collaborate effectively with teams and stakeholders following alternative approaches.**

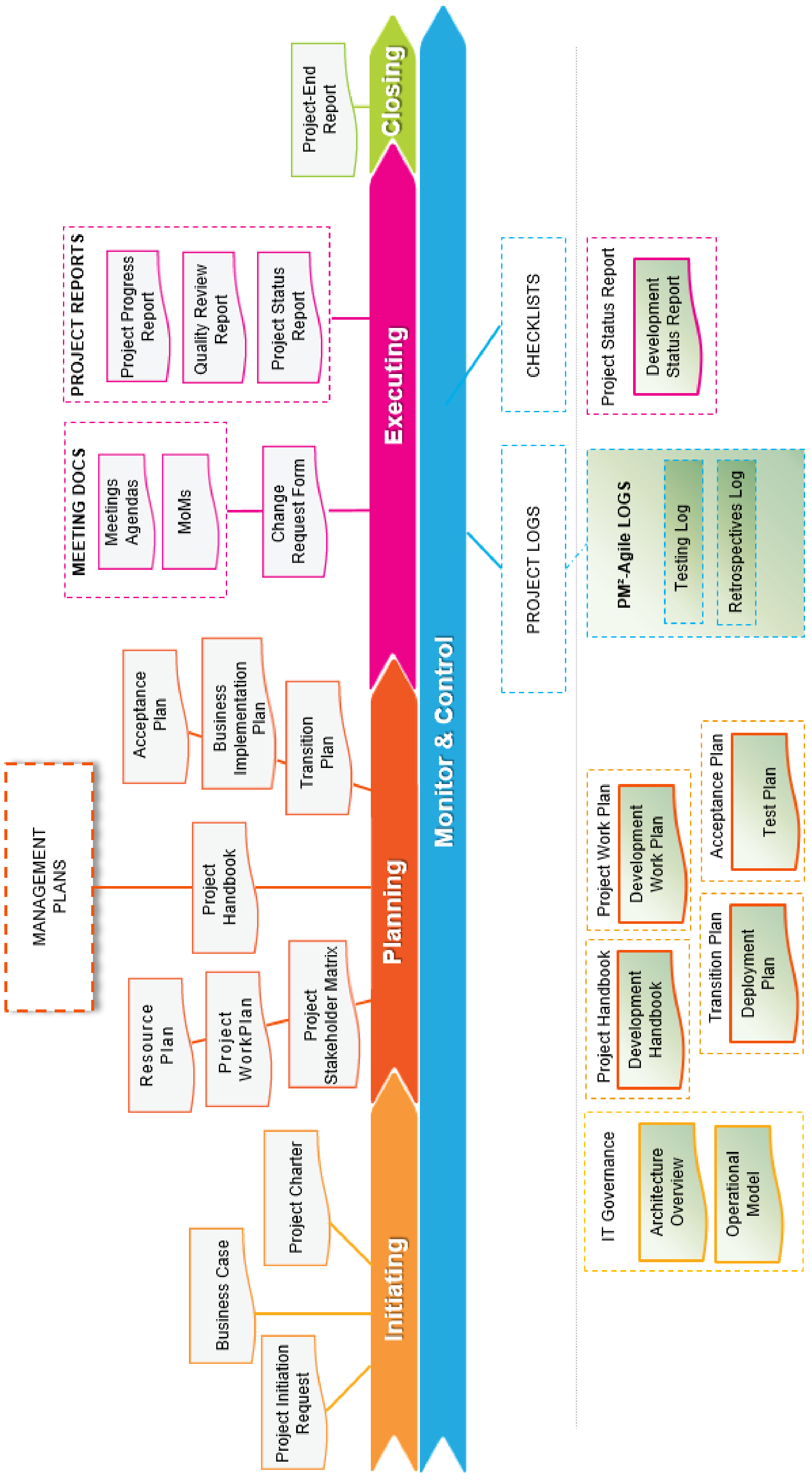
When practising Agile in big organisations, there is a need to enable collaboration between Agile and other project approaches while complying with various IT governance, enterprise architecture, interoperability, and operational requirements. Failing to do so can bring, at best, local successes for a short period of time, and not sustainable solutions, an essential part of the overall organisation.

## B.2 Agile Artefacts and Ceremonies Summary Tables and Diagrams

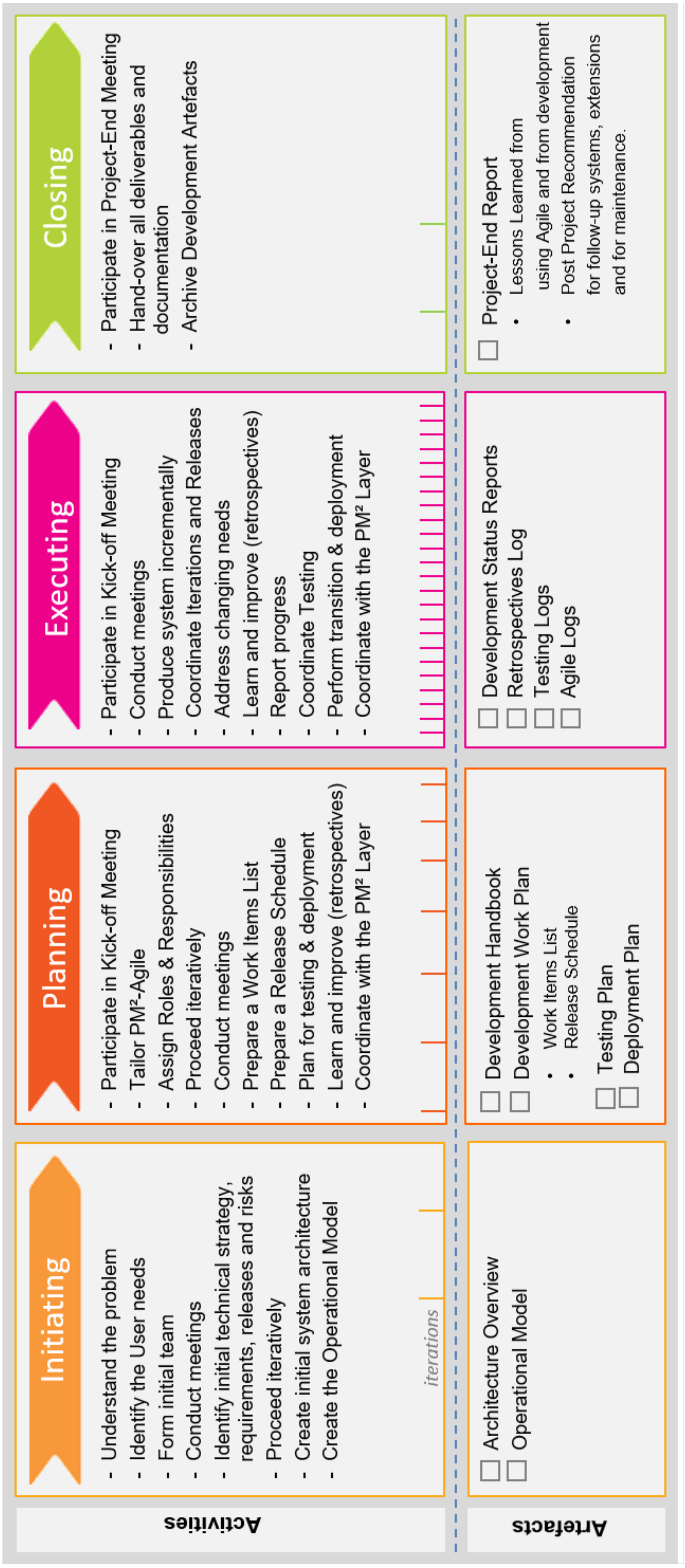
Note: RAM (RASCI) (Responsible, Accountable, Supports, Consulted, Informed)

| **Initiating** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
|---|---|---|---|---|---|---|
| Business Case | R | S | - | - | - | - |
| Project Charter | S | R | - | - | - | - |
| Architecture Overview | I | A | S | I | R | S |
| Operational Model | I | A | S | I | R | S |
| **Planning** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
| Development Handbook | I | A | R | S | S | S |
| Development Work Plan | I | A | R | S | S | S |
| • Work Items List | S | A | C | R | S | S |
| • Release Plan | I | A | R | S | S | S |
| • Iteration Plan | I | A | R | C | S | S |
| Test Plan | I | A | S | S | S | R |
| Deployment Plan | I | A | R | S | S | S |
| Iteration Planning meeting | I | I | S | S/A | S | R |
| **Executing** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
| Development Status Report | I | A | R | I | I | S |
| Retrospectives Log | I | A | R | I | I | S |
| Testing Logs | I | A | R | I | S | S |
| Daily Stand-up meeting | N/A | I | S | I | R | R/A |
| Iteration Review meeting | C | I | S | S/A | R | R |
| Iteration Retrospective meeting | I | I | S | I | R | R/A |
| **Monitor & Control** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
| Release Planning | | | | | | |
| • Manage Work Items List | S | I | S | R/A | I | I/S |
| • Work Items List refinement | I | I | S | R | I | S/A |
| • Work Items List prioritisation | S | I | S | R/A | C | C |
| **Closing** | **BM** | **PM** | **TeCo** | **PrOw** | **ArOw** | **ATeM** |
| Project-End Report | S | R | S | I | I | I |

## B.3: The PM² and PM²-Agile Artefacts Landscape



## B.4: Overview of PM²-Agile Activities & Artefacts



| | Initiating | Planning | Executing | Closing |
|---|---|---|---|---|
| Activities | - Understand the problem<br>- Identify the User needs<br>- Form initial team<br>- Conduct meetings<br>- Identify initial technical strategy, requirements, releases and risks<br>- Proceed iteratively<br>- Create initial system architecture<br>- Create the Operational Model<br>*iterations* | - Participate in Kick-off Meeting<br>- Tailor PM²-Agile<br>- Assign Roles & Responsibilities<br>- Proceed iteratively<br>- Conduct meetings<br>- Prepare a Work Items List<br>- Prepare a Release Schedule<br>- Plan for testing & deployment<br>- Learn and improve (retrospectives)<br>- Coordinate with the PM² Layer | - Participate in Kick-off Meeting<br>- Conduct meetings<br>- Produce system incrementally<br>- Coordinate Iterations and Releases<br>- Address changing needs<br>- Learn and improve (retrospectives)<br>- Report progress<br>- Coordinate Testing<br>- Perform transition & deployment<br>- Coordinate with the PM² Layer | - Participate in Project-End Meeting<br>- Hand-over all deliverables and documentation<br>- Archive Development Artefacts |
| Artefacts | ☐ Architecture Overview<br>☐ Operational Model | ☐ Development Handbook<br>☐ Development Work Plan<br>• Work Items List<br>• Release Schedule<br>☐ Testing Plan<br>☐ Deployment Plan | ☐ Development Status Reports<br>☐ Retrospectives Log<br>☐ Testing Logs<br>☐ Agile Logs | ☐ Project-End Report<br>• Lessons Learned from using Agile and from development<br>• Post Project Recommendation for follow-up systems, extensions and for maintenance. |

## B.5: References and further Reading

**Title:** The Agile Manifesto

**Author:** Various Authors

**Publication Date:** 2001

**URL:** http://Agilemanifesto.org/

**Title:** The Scrum Guide

**Author:** Jeff Sutherland & Ken Schwaber

**Publisher:** Scrum.org

**URL:** http://www.scrum.org/Scrum-Guides

**Title:** Succeeding with Agile

**Author:** Mike Cohn

**Publisher:** Addison-Wesley

**Publication Date:** November 2009

**ISBN:** 0321579364

**Title:** Kanban: Successful Evolutionary Change for Your Technology Business

**Author:** David J Anderson

**Publication Date:** April 2010

**ISBN:** 0984521402

**Title:** The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win

**Author:** Gene Kim, Kevin Behr, George Spafford

**Publisher:** IT Revolution Press

**Publication Date:** 10 January 2013

**ISBN:** 0988262592

**Title:** Essential Scrum: A Practical Guide to the Most Popular Agile Process

**Author:** Kenneth S. Rubin

**Publisher:** Addison-Wesley Professional

**Publication Date:** 5 August 2012

**ISBN:** 0988262592

**Title:** Agile Software Development, Principles, Patterns, and Practices

**Author:** Robert C. Martin

**Publisher:** Prentice Hall

**Publication Date:** 25 October 2002

**ISBN:** 0135974445

**Title:** Lean Software Development: An Agile Toolkit

**Author:** Mary Poppendieck, Tom Poppendieck

**Publisher:** Addison-Wesley

**Publication Date:** May 2003

**ISBN:** 0321150783

**Title:** The Lean Startup

**Author:** Eric Ries

**Publisher:** Portfolio penguin

**Publication Date:** 2011

**ISBN:** 0670921602

**Title:** Lean UX

**Author:** Jeff Gothelf and Josh Seiden

**Publisher:** O'Reilly

**Publication Date:** 2016

**ISBN:** 9781491953600

**Title:** Lean vs Agile vs Design Thinking

**Author**: Jeff Gothelf

**Publisher:** Gothelf Group

**Publication Date**: 2017

**ISBN:** 13:978-1541140035

**Title:** Sprint- how to solve big problems and test new ideas in just five days

**Author:** Jake Knapp, John Zeratsky and Braden Kowitz

**Publication Date:** 2016

**ISBN:** 9780593076118

### Additional reading

**Title:** Agile Project Management using Scrum

**Author:** Ken Schwaber

**Publisher:** Microsoft Press

**Publication Date:** March 2004

**ISBN:** 073561993X

**Title:** Balancing Agility and Discipline

**Author:** Barry Boehm and Richard Turner

**Publisher:** Addison-Wesley

**Publication Date:** August 2003

**ISBN:** 0321186125

**Title:** Extreme Programming Explained: Embrace Change (2nd Edition)

**Author:** Kent Beck / Cynthia Andres

**Publisher:** Addison-Wesley

**Publication Date:** Nov 2004

**ISBN:** 978-0321278654

**Title:** An Agile Adoption and Transformation Survival Guide

**Author:** Michael Sahota

**Publisher:** InfoQ

**Publication Date:** 2012

**ISBN:** 978-1-1-5-73572-1

**URL:** http://www.infoq.com/minibooks/Agile-adoption-transformation

**Title:** Kanban and Scrum — making the most of both

**Author:** Henrik Kniberg and Mattias Skarin

**Publisher:** InfoQ

**Publication Date:** December 2009

**ISBN:** 978-0-557-13832-6

**URL:** http://www.infoq.com/minibooks/kanban-scrum-minibook

**Title:** The Scrum Primer

**Author:** Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde

**Publisher:** InfoQ

**Publication Date:** December 2009

**URL:** http://www.infoq.com/minibooks/Scrum_Primer

**Title:** The Culture Game: Tools for the Agile Manager

**Author:** Daniel Mezick

**Publisher:** InfoQ

**Publication Date:** October 2012

**ISBN:**

**URL:** http://www.infoq.com/minibooks/Mezick-Culture-game

**Title:** Scrum and XP from the Trenches

**Author:** Henrik Kniberg

**Publisher:** InfoQ

**Publication Date:** June 2007

**ISBN:** 978-1-4303-2264-1

**URL:** http://www.infoq.com/minibooks/scrum-xp-from-the-trenches

**Title:** The Mythical Man Month (2nd Edition)

**Author:** Frederick P Brooks

**Publisher:** Addison-Wesley

**Publication Date:** August 1995

**ISBN:** 0201835959

**Title:** Leading Change

**Author:** John P Kotter

**Publisher:** Harvard Business School Press

**Publication Date:** November 2012

**ISBN:** 1422186431

## B.6: Other Useful Resources

**The PM² Guide (pdf)**

This guide documents the PM² project management methodology. It has been kept as light as possible but still provides enough information to help you understand and use PM² effectively. You can download the pdf version of the guide from the PM² website: https://europa.eu/pm2/pm2-material_en

**The PM² Programme Guide (pdf)**

**Link to PM²-Agile on the PM² Website:** https://europa.eu/pm2/pm2-agile_en

**Link to CoP:** https://webgate.ec.europa.eu/fpfis/wikis/x/pIB3Bw

This page has intentionally been left blank

# Appendix C: Glossary

| A | |
|---|---|
| Acceptance Criteria | Those criteria by which a work item can be evaluated to have been implemented according to the client's expectations. Most commonly, these criteria are expected to be 'all or nothing' – that is, either all criteria pass and the work item is a candidate to be 'done', or the work item is not 'done'.<br>See also Acceptance Testing. |
| Acceptance Testing | Acceptance testing is the final test action before releasing a solution. The goal of acceptance testing is to verify that the solution is ready and can be used by the users to perform those functions and tasks for which the solution was built. |
| Agile Development | Agile development is an approach to user interface and software development following the Agile Manifesto. It is currently applied by several methodologies based on Iterative Development, where requirements and solutions evolve through collaboration between self-organising, cross-functional teams. |
| Agile Estimating | Agile estimating is based on an iterative approach, in which the estimates improve as the project evolves and the estimators have a better understanding of what is being estimated. Most Agile estimation techniques use relative units instead of time units. |

| B | |
|---|---|
| Backlog | Also known as 'Product Backlog'. See Work Items List (WIL), the PM²-Agile term for Product Backlog. |
| Branching | Branching is the duplication of objects under revision control (such as a source code file, or a directory tree) in such a way that the newly created objects initially have the same content as the original, but can evolve independently of the original.<br>Branching can take two forms, static or dynamic. In static branches, copy and label operations are used to duplicate a given branch. The duplicate then can evolve independently. With dynamic branches, usually implemented in streams, only the label operation is used, to flag the point in time that a stream diverged from its parent stream. Both branching forms support some form of merging, so that code changes made on a branch can be re-integrated into another branch, as is typical in parallel development processes. |
| Burndown Chart | Graphical representation of work left to do versus time. The outstanding work (Work Items List) is often on the vertical axis, with time along the horizontal. |
| Burnup Chart | Representation of the Work Items completed; usually represented in chart form with points plotted on an x and y axis that map an upward trend of work completed until reaching 100%. |

| C | |
|---|---|
| Change and Configuration Management | Change and Configuration Management (also known as Software Change and Configuration Management (SCCM)) combines aspects of both change management and Configuration Management to control a software development project as it evolves through the software development process. SCCM typically includes all technical aspects of the development process, such as version control, branching and merging. |

| | |
|---|---|
| | Additionally, SCCM includes change-related activities such as issue tracking, document tracking, and process workflows that enable development teams to control the overall process. |
| Collocation | Collocation refers to development teams located and working in the same location. Collocation is usually applied at the cross-functional team level. |
| Continuous Integration | Continuous Integration, one of the foundational aspects of Agile software development methodologies, is defined by Martin Fowler to be 'a fully automated and reproducible build, including testing, that runs many times a day. This allows each developer to integrate daily, thus reducing integration problems.' By getting changes into the main line as frequently as possible, preferably daily, and by extending the idea of a nightly build, Continuous Integration helps reduce integrations problems and identify and resolve problems more quickly. |
| Cross-Functional Team | Team comprised of members with all functional skills and specialties necessary to complete a project from start to finish. |

| **D-Z** | |
|---|---|
| Design thinking | Design Thinking is an iterative process that combines various techniques and tools to solve problems creatively in a holistic manner and enhances innovation by focusing on the real user's needs. |
| Iteration | Predefined, timeboxed and recurring period of time in which the working solution is created. The most commonly used iteration durations range from one to four weeks. |
| INVEST model | INVEST stands for "Independent, Negotiable, Valuable, Estimable, Small and Testable" and is a set of criteria used to assess the quality of a user story. |
| Kanban | Methodology that comes from Lean software development and has three main aspects: it uses a visual system for managing work, it limits work in progress, and work is 'pulled' rather than 'pushed' through the system. |
| Key Agile Principles | See Agile Manifesto. |
| Lean Software Development | Lightweight approach to software development, based on Lean Manufacturing. According to Mary Poppendieck, the 10 rules of Lean programming are: eliminate waste, minimise artefacts, satisfy all stakeholders, deliver as fast as possible, decide as late as possible, decide as low as possible, deploy comprehensive testing, learn by experimentation, measure business impact and optimise across organisations. |
| Merging | The process of incorporating branches back into the mainline. |
| MoSCoW Prioritisation | MoSCoW is a prioritization technique used to reach a common understanding on the importance of delivering of each requirement; The acronym, MoSCoW, stands for: Must-have, Should-have, Could-have, and Want to have (but won't have this time). |
| Pair Programming | Process in which two developers work together at a single workstation, where one is responsible for typing code and the other for reviewing each line of code as it is typed in. |
| Parallel Development | Parallel development occurs whenever a software development project requires separate development efforts on related code bases. For example, when a software product is shipped to customers, a product development team may begin working on a new major feature release of the product, while a product maintenance team may work on defect correction and customer patch releases of the shipped product. Both teams begin working from the same code base but the |

| | |
|---|---|
| | code will necessarily diverge. The code bases used in parallel development efforts must be merged frequently to ensure, for example, that the defect corrections provided by the product maintenance team are integrated into the major release the product development team is working on. |
| Planning Poker | A consensus-based technique for estimating; mostly used to estimate relative size of tasks in software development. Planning Poker is useful for building team cohesion and for fostering self-organising teams. |
| Product Backlog | See Backlog. The equivalent in PM²-Agile is the Work Items List. |
| Refactoring | The practice of continuously improving the usability, maintainability, and adaptability of code without changing its behaviour. Refactoring makes it much easier to add new and unanticipated functionality. Refactoring has the disadvantage that it takes extra effort and requires changing the code. |
| Release Management | Release management comprises a broad set of activities in software development organisations that focus on ensuring that software is ready to be released to customers. |
| Release Process | The release process is the final stage, where the solution is made available for use. The release process must ensure that all product requirements have been met, usually by executing test suites designed to exercise product functionality and correcting any defects found. |
| Release Schedule | The Release Schedule documents the high-level development milestones. It represents the part of the Development Work Plan. |
| Scrum | Agile development project management framework based around sprints and is generally comprised of a Scrum Development Team, Product Owner and Scrum Master. The framework of Scrum leaves most development decisions up to the self-organising Scrum team, where consensus is reached as a whole team. |
| Story Points | Relative scale of effort required by a team to implement a story. |
| Test-Driven Development (TDD) | Test-Driven Development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards. |
| Timeboxing | The practice of constraining the amount of time for performing any activity. Examples include iterations, and stand-up meetings. |
| Unit Testing | Tests that exercise small amounts of isolated functionality. |
| User Stories | Used with Agile methodologies for specifying and presenting requirements as an informal statement (usually fitting on a 3x5 inch index card). |
| Velocity | The velocity of a team is the number of story points associated with stories that are finished over a timeboxed iteration. |
| Version Control | Version control is a practice focused on managing the changes made to a set of configuration items that are part of an information system, such as code, documentation, build scripts, etc. |
| Waterfall | Model of a software development process in which progress flows downwards through phases of conception, initiation, analysis, design, construction, testing and maintenance. |

| | |
|---|---|
| Work Board | A physical or electronic board representing the state of Work Items in a current iteration, often divided into 'backlog', 'to do,' 'doing' and 'done.' |
| Work Item | A piece of work that needs to be built in order to contribute to a solution. It can be a user story, bug, enabler story, etc. The set of Work Items that represent the solution to be built is the Work Items List (WIL). |

NO-02-21-822-EN-N

PM² - Agile
Guide 3.0.1

CoEPM²
European Commission
Centre of Excellence in PM²

Publications Office
of the European Union

ISBN 978-92-76-39410-5