Certified Professional for Requirements Engineering

RE@Agile

Syllabus

Stefan Gärtner, Peter Hruschka,
Markus Meuten, Gareth Rogers,
Hans-Jörg Steffe

2.3.0 | October 01, 2025

Terms of use:

1. Individuals and training providers may use this syllabus as a basis for seminars, provided that the copyright is acknowledged and included in the seminar materials. Anyone using this syllabus in advertising needs the written consent of IREB e.V. for this purpose.
2. Any individual or group of individuals may use this syllabus as basis for articles, books or other derived publications provided the copyright of the authors and IREB e.V. as the source and owner of this document is acknowledged in such publications.

Acknowledgements

The current version of this syllabus has been written by: Stefan Gärtner, Peter Hruschka, Kim Lauenroth, Markus Meuten, Gareth Rogers and Hans-Jörg Steffe.

Review by Rainer Grau. Review comments were provided by Jan Jaap Cannegieter, Andrea Herrmann, Uwe Valentini and Sven van der Zee. English review by Gareth Rogers.

Approved for release on October 25, 2024 by the IREB Council upon recommendation of Xavier Franch.

We thank everybody for their involvement.

Purpose of the document

This syllabus defines the certificates for the RE@Agile Practitioner and for the RE@Agile Specialist of the International Requirements Engineering Board (IREB). The syllabus provides training providers with the basis for creating their course materials. Students can use the syllabus to get an overview of the intended content and the learning objectives. Content details for the preparation of the course material and for the exam can be found in the "Handbook RE@Agile, Education and Training for the IREB Certified Professional for Requirements Engineering RE@Agile Practitioner | Specialist".

Contents of the syllabus

The RE@Agile module is aimed at people from Requirements Engineering, Business Analysis, Business Engineering, Software and System Development as well as existing roles in the agile community (Product Owner, Scrum Master, Developer) who would like to deepen their knowledge in dealing with requirements.

Level of detail

The level of detail of this syllabus allows internationally consistent teaching and examination. To reach this goal, the syllabus contains the following:

- General educational objectives,
- Contents with a description of the educational objectives, and
- References to further literature (where necessary).

Educational objectives / cognitive knowledge levels

All modules and educational objectives in this syllabus are assigned a cognitive level. The following levels are used:

- **L1: Know** (identify, remember, retrieve, recall, recognize) – The candidate will recognize, remember and recall a term or concept.
- **L2: Understand** (summarize, generalize, abstract, classify, compare, map, contrast, exemplify, interpret, translate, represent, infer, conclude, categorize, construct models) – The candidate can select the reasons or explanations for statements related to the topic, and can summarize, compare, classify, categorize and give examples for the concept.
- **L3: Apply** (implement, execute, use, follow a procedure, apply a procedure) – The candidate can select the correct application of a concept or technique and apply it to a given context.
- **L4: Analyze** (analyze, organize, find coherence, integrate, outline, parse, structure, attribute, deconstruct, differentiate, discriminate, distinguish, focus, select) – The candidate can separate information related to a procedure or technique into its constituent parts for better understanding, and can distinguish between facts and inferences. Typical application is to analyze a document, software or project situation and propose appropriate actions to solve a problem or task.
- **L5: Evaluate** (critique, judge) – The candidate can give a well-argued critique of a given artifact, and make a profound judgment in a given case.

Note that a learning objective at cognitive knowledge level Ln also contains elements of all cognitive levels below it (L1 to Ln-1).

> Example: A learning objective of the type "Apply the RE technique xyz" is at the cognitive knowledge level (L3). However, the ability to apply requires that learners know RE technique xyz (L1) and that they understand what the technique is for (L2).

All terms used in this syllabus and defined in the IREB Glossary have to be known (L1), even if they are not explicitly mentioned in the educational objectives.

The glossary is available for download on the IREB website at https://www.ireb.org/en/downloads/#cpre-glossary-2-0

This syllabus and the related handbook use the abbreviation "RE" for Requirements Engineering.

Structure of the syllabus

The syllabus consists of six main chapters. One chapter covers one educational unit (EU). Chapter titles contain the cognitive level of their chapters, which is the highest level of their sub-chapters. The teaching time suggested is the minimum a course should invest for that chapter. Training companies are free to devote more time to the EUs and the exercises. However, they should ensure that the time required is maintained in relation to the other EUs. Important terms of each chapter are listed at the beginning of the chapter.

Example: EU2 A Clean Project Start (L2)
Duration: 120 minutes + 60 minutes exercise
Terms: Product Vision, Product Goal, Stakeholder, Persona, Product Scope, System Boundary

This example shows that chapter 2 contains education objectives at level L2, and 180 minutes are intended for teaching the material in this chapter.

Each chapter can contain sub-chapters. Their titles also contain the cognitive level of their content.

Educational objectives (EO) are enumerated. The numbering shows to which sub-chapter they belong.

Example: EO 3.3.1 Be able to apply the INVEST criteria when writing requirements (L3)

This example shows that educational objective EO 3.3.1 is described in sub-chapter 3.3 and that cognitive level L3 is expected.

The examination

This syllabus covers educational units and educational objectives for the certification exams of the

- CPRE RE@Agile – Practitioner
- CPRE RE@Agile – Specialist

The exam to obtain the CPRE RE@Agile – Practitioner – certificate consists of a **multiple-choice exam**.

The exam to obtain the CPRE RE@Agile – Specialist – certificate consists of a **written assignment**.

Both exams include exam questions on all educational units and all educational objectives of the syllabus.

Each exam question may include material from multiple chapters of the syllabus as well as multiple learning objectives or even from parts of one learning objective.

The multiple-choice exam for the Practitioner certificate

- tests all learning objectives of the syllabus. However, for the educational objectives at cognitive knowledge levels L4 and L5, the exam questions are limited to items at cognitive levels L1 through L3.
- can be held immediately after a training course, but also independently from courses (e.g., remote or in an examination center).

The written assignment for the Specialist certificate

- tests all educational objectives of the syllabus at the cognitive knowledge levels indicated for each learning objective.
- follows the task description for RE@Agile – Specialist –, found at https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-re-agile-specialist-written-assignment.
- is self-paced and submitted to a licensed Certification Body.

The following generic learning objectives also apply to the written assignment for the Specialist certificate:

EO G1: Analyze and illustrate RE@Agile problems using a context familiar to the candidate or one that closely resembles it (L4).

EO G2: Evaluate and reflect on the usage of RE@Agile practices, methods, processes and tools in projects that the candidate was involved in (L5).

A list of IREB licensed certification bodies can be found on the website:

https://www.ireb.org

Version history

Version	Date	Comment	Author
1.0.0	February 20, 2018	Initial Version	Bernd Aschauer, Lars Baumann, Peter Hruschka, Kim Lauenroth, Markus Meuten, Sacha Reis and Gareth Rogers
1.0.1	September 11, 2018	Typos fixed A couple of EO's reformulated to meet standard style. No change content wise. Statement on important terms clarified	Peter Hruschka, Stefan Sturm
1.0.2	September 24, 2018	Credits for reviewers added	Stefan Sturm
1.0.3	December 17, 2019	Consistent usage of the term refinement meeting and product backlog refinement	Hans-Jörg Steffe
2.0.0	July 1, 2022	Educational objective levels updated New version of chapter 6 Corrections in all chapters Rework on the suggested teaching time and practice time on all chapters Information about Advanced Level exam split added.	Peter Hruschka, Markus Meuten, Gareth Rogers, Stefan Gärtner, Hans-Jörg Steffe
2.1.0	May 1, 2024	New Corporate Design implemented, Cognitive Knowledge Levels synchronized, Term "Advanced Level removed".	Stan Bühne
2.2.0	May 24, 2024	Cognitive Knowledge Levels fixed again.	Stefan Sturm

2.3.0	October 1, 2025	Learning objectives updated and complemented. Short description of the chapters rewritten Added illustrations to the chapters	Hans-Jörg Steffe

Content

1 What is RE@Agile (L2)

Duration: 45 minutes

Terms: Stakeholder, Product Owner, Cooperative, Iterative, Incremental

Educational objectives

EO 1.1 Know the definition of RE@Agile (L1)
EO 1.2 Understand the goals of RE@Agile (L2)
EO 1.3 Understand that the responsibility for good requirements is with the product owner (L2)

Content

Based on their respective histories, RE and agile approaches are often considered separately rather than together. This often leads to the misunderstanding that there are two ways of RE: classical RE and agile RE. The authors believe that there is only good or bad RE – in a non-agile or agile world. Therefore, we call the approach RE@Agile.

Agile and RE are two disciplines with different origins and distinct goals that can nevertheless learn a lot from each other. In the RE@Agile Primer [Prim2017] we concluded: "The most important value is shared by RE and agile, and that is to make the user of the product happy because the solution fits their needs or cures their greatest pains."

RE@Agile is a cooperative, iterative and incremental approach with four goals:

1. Knowing the relevant requirements at an appropriate level of detail (at any time during system development).
2. Achieving sufficient agreement about the requirements among the relevant stakeholders.
3. Capturing (and documenting) the requirements according to the constraints of the organization.
4. Performing all requirements-related activities according to the principles of the Agile Manifesto.

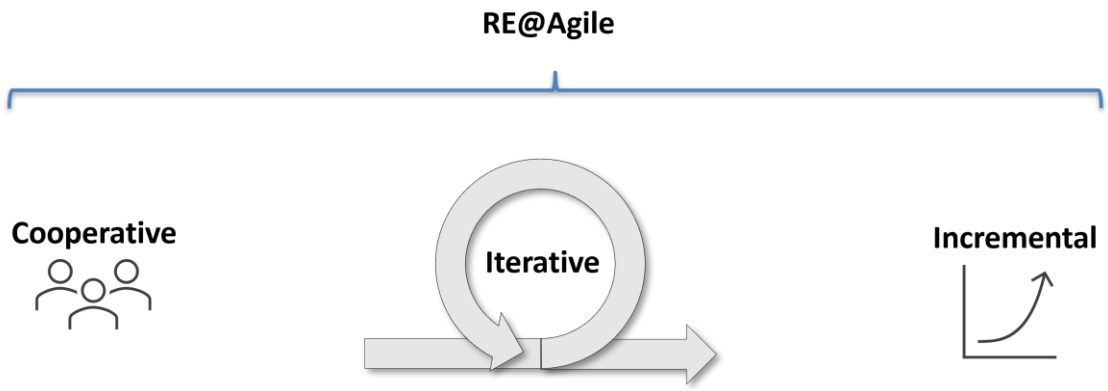

Figure 1: Cooperative, iterative and incremental approach

In agile approaches, different roles provide requirements for the desired system. Regardless of who delivers these requirements, who structures them, and who details them, the person with the role/responsibility of product owner remains responsible for RE.

2 A clean project start (L3)

Duration: 120 minutes + 60 minutes exercise

Terms: Vision, Goal, Stakeholder, System boundary, Context diagram, Use case diagram

Content

Even in agile approaches some important prerequisites have to be established before successful iterative and incremental system development work can start.

- Definition of the vision and/or goals of the system
- Identification of the currently known scope of the system and the system boundary
- Identification of relevant stakeholders and other important requirements sources

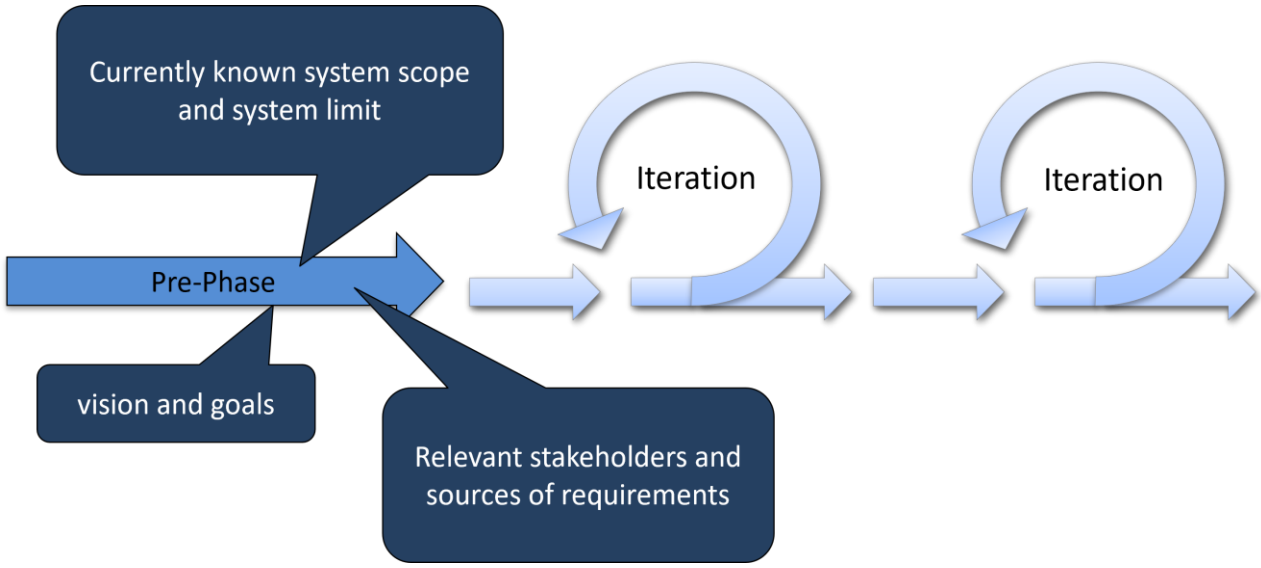

Figure 2: A clean project start

2.1 Vision and goals (L3)

Educational objectives

EO 2.1.1 Apply the specification of goals and visions (L3)

Content

The system vision or product vision describes the overall goal that shall be achieved with the system/product. The vision is of utmost importance for every development activity. It defines the cornerstone and serves as an overall direction for all development activities. Every requirement should support achieving the system vision.

Alternative approaches to the formulation of goals or visions are:

- SMART [Dora1981]
- PAM [Robe2003]
- Product Vision Box [High2001]
- News from the future [HeHe2011]

- Vision Boards
- Canvas Techniques

2.2 Specifying the system boundary (L3)

Educational objectives

EO 2.2.1 Apply the specification of the system boundary (L3)

Content

A shared and common understanding of the scope and the context of the system is a prerequisite for an effective and efficient development effort [Glin2024].

The scope and the system boundary can be documented and clarified with several techniques, such as:

- Context diagrams
- Natural language
- Use case diagrams
- Story maps

2.3 Stakeholder identification and management (L3)

Educational objectives

EO 2.3.1 Mastering and using stakeholder identification and management (L3)

Content

Similar to traditional approaches, the most important stakeholders must also be identified at the beginning of the agile process so that a framework is set for the requirements elicitation. In agility, however, the stakeholders can and will change constantly. It is therefore essential that the identification and definition of stakeholders is itself understood as a cooperative, incremental and, above all, iterative process (see also chapter 1).

Failing to identify and involve an important stakeholder in the development process can have a major impact. If the requirements of such a stakeholder are identified too late (or not at all), this may result in costly changes or even a useless system [PoRu2021].

The Onion Model from Ian Alexander [Alex2005] is a simple tool for stakeholder identification and classification. The model consists of three types of stakeholders (onion layers) that can be systematically searched for stakeholders:

- Stakeholders of the system
- Stakeholders of the surrounding context
- Stakeholders of the wider environment

As a rule of thumb, the identification of stakeholders should rely on a broad range of sources.

Depending on the system and the domain, existing documentation, neighboring systems with interfaces to the developed system, legacy systems or even competitor systems may also be important sources (in addition to stakeholders) of requirements.

2.4 Dependencies between visions/goals, stakeholders and the system boundary (L3)

Educational objectives

EO 2.4.1 Apply the dynamic change of vision and goals, stakeholders and system scope (L3)

Content

The definitions of vision and goals as well as stakeholders and system boundaries are here interdependent [PoRu2021]:

- Relevant stakeholders formulate the vision and the goals. Therefore, the identification of a new relevant stakeholder may have an impact on the vision and goals.
- Vision and goals can be used to guide the identification of new stakeholders by asking: Which stakeholder may be interested in achieving the vision and goals or is affected by achieving them?
- Vision and goals can be used to define an initial scope by asking: Which elements are necessary to achieve the vision and goals?
- Changing the system boundary (and thus the scope) may have an impact on the vision and goals. If an aspect is removed from the scope, it has to be verified that the system still has sufficient means to achieve the vision and goals.
- Stakeholders define the system boundary. Therefore, the identification of a new, relevant stakeholder may have an impact on the scope.
- A change of the scope (e.g., to fulfill a goal) requires agreement from the relevant stakeholders.

These interdependencies should be used to balance all three elements (vision and goals, stakeholders, scope) to examine the impact of changing one of the three elements on the other.

Because of these dependencies between vision and goals, stakeholders, and scope we recommend treating all these elements together and in a coherent way.

3 Handling functional requirements (L4)

Duration: 195 minutes + 120 minutes exercise

Terms: Functional requirement, Epic, Feature, Story, User story, Definition of Ready (DoR), INVEST, Levels of granularity

Content

This core EU takes a static view on functional requirements, i.e. structuring a large set of requirements into abstraction hierarchies.

As soon as the idea has been accepted that requirements do exist on different levels of granularity, some questions naturally arise:

- How do we deal with multiple levels of granularity?
- Which criteria can and should be applied to split big, abstract topics into smaller chunks?
- Is it sometimes necessary to group many small requirements into larger chunks so that we have a "bigger picture" for orientation?
- How precise do we have to be before the developers can begin with the implementation?
- Is it necessary or advisable to keep multiple levels of requirements, or can we throw away abstract statements as soon as we have more concrete requirements?
- Do we prefer to structure the backlog according to functional relations/processes, or according to other relations such as technical contexts?
- Do we have to capture all of this in writing or can we simply talk about it?

3.1 Different levels of requirements granularity

Educational objectives

EO 3.1.1 Know the existence of functional requirements at different levels of granularity (L1)

Content

Stakeholders usually communicate requirements on different levels of granularity. Sometimes they ask for big chunks of functionality, sometimes they ask for minor details to be added or changed.

Epics (sometimes also called themes), or large stories (representing potentially complex business processes) are a good way to get a big picture of all currently known requirements in the backlog, i.e., an overview of all the things that stakeholders expect from a system or a product. Epics are decomposed into finer-grained elements, typically features and stories. Stories provide the finer level of granularity and may communicate requirements from different perspectives, e.g., user perspective, technical perspective.

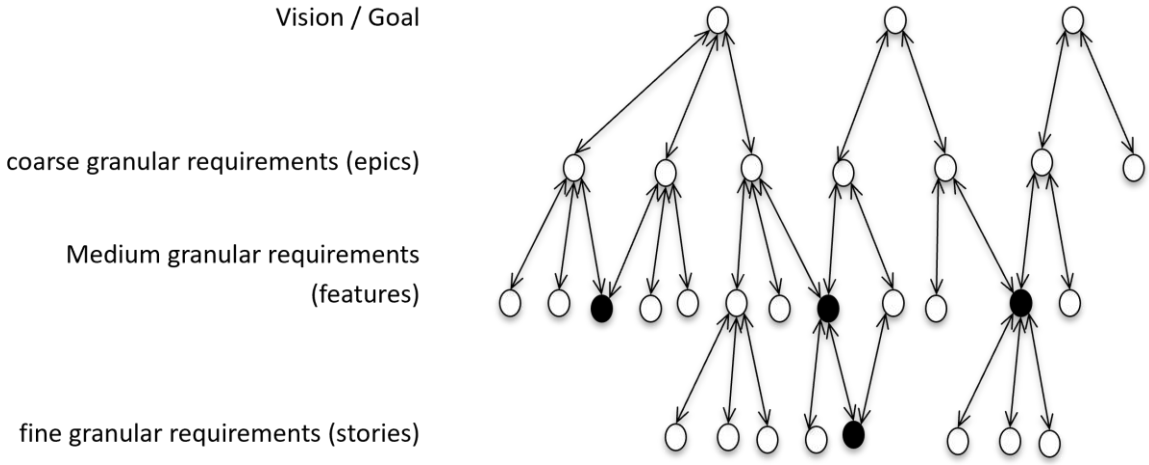

Figure 3: Requirements granularity

3.2 Communicating and documenting on different levels

Educational objectives

EO 3.2.1 Analyze and apply the decomposition of requirements at the highest level (L4)
EO 3.2.2 Analyze and apply different decomposition strategies in the large (L4)
EO 3.2.3 Analyze and apply the identification, documentation, and communication of functional requirements on different levels of granularity (L4)

Content

Based on the principle of "divide and conquer", we need to decompose a large system or product into smaller parts.

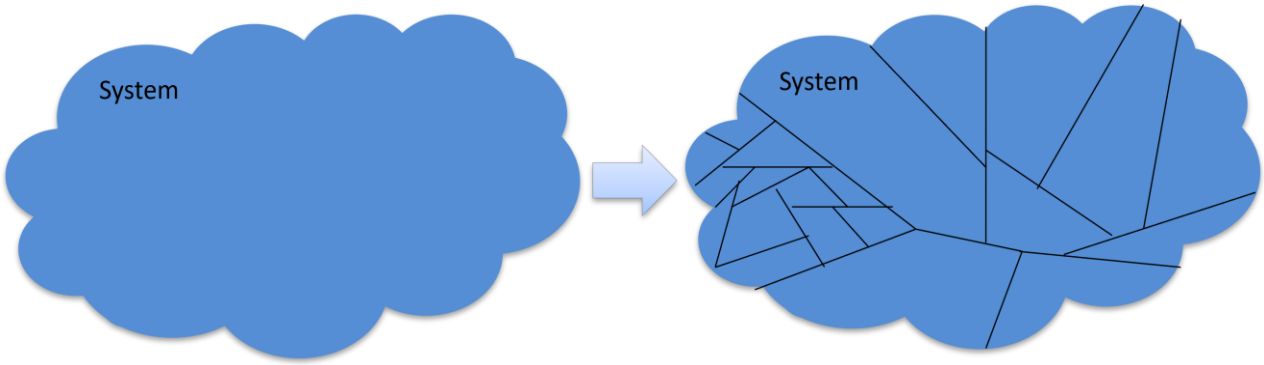

Figure 4: System decomposition into smaller parts

There are different approaches to achieving this goal (e.g., division into logical functions, division according to organizational aspects, division according to hardware, division according to data, ...). In doing so, we look at the approach to breaking down requirements into externally initiated, value-creating processes.

In addition to decomposing the requirements into smaller parts, we also have to take care of communicating and documenting the functional requirements. The basic choice is between

drawing (e.g., with models) and writing (e.g., in natural language). Above all, it is up to the whole team to decide which format is preferred. For example, the overview could be provided using a use case diagram, while the process steps of the use cases are described in simple words (e.g., as a user story).

3.3 Working with user stories and backlog items

Educational objectives

EO 3.3.1 Apply the idea of User Centered Thinking on requirements with User Stories (L3)
EO 3.3.2 Create requirements based on the user story idea and using the user story template (L3)
EO 3.3.3 Apply the INVEST criteria when writing requirements (L3)

Content

User stories have become very important in the agile environment because they point to the importance of structuring and developing requirements and the resulting products in a user-centered way. In other words, putting the user first. This user perspective is important because we primarily develop in an agile way in order to receive regular feedback from users.

It is important to remark that in the agile environment, the term "user story" is often used with three different purposes:

- As a user-oriented way to discuss and formulate requirements.
- As the lowest (most detailed) level of requirements structuring (Epics -> Features -> (User-) Stories). In fact, the term "user story" is often used instead of "story" in the requirements structure.
- As a template to describe requirements at different hierarchy levels from a user's perspective.

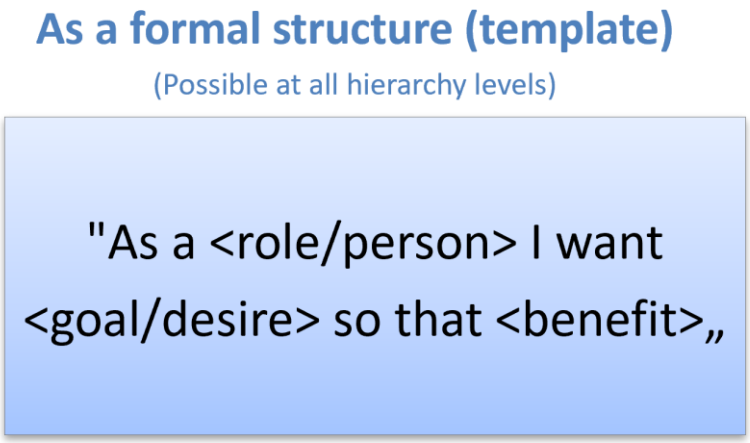

Figure 5: User story template

In this syllabus, in order to mitigate this ambiguity, we use the term "story" for the finer-grained elements in the requirements structure; therefore, a "user story" is a particular type of story, i.e. a user-centered story. When we refer to the template presented in Fig. 5, we will use the term "user story template". Please note that the user story template can be used to describe not only stories but also epics and features if deemed appropriate.

In general, it should be noted that user stories are not complete requirements in themselves; they are more of a communication promise. In order to create complete requirements and corresponding backlog items, further detailing (documented and/or in discussion) will have to take place.

When introducing user stories [Cohn2010], Bill Wake [Wake2003] defined that user stories should comply with the INVEST principles (Independent, Negotiable, Valuable, Estimated, Small, Testable).

Backlog items are often written on index cards or sticky notes in addition to IT tools (e.g., Jira MS-Devops) and arranged on walls or tables to facilitate planning and discussion. Using the 3C model (card, conversation, confirmation), these backlog items can then be discussed in more detail and supplemented with other requirement artifacts (e.g., models, documentation).

3.4 Splitting and grouping techniques

Educational objectives

EO 3.4.1 Analyze and apply splitting techniques for coarse-grained functional requirements (L4)
EO 3.4.2 Analyze and apply grouping and abstraction of detailed functional requirements into coarser requirements (L4)

Content

In order to generate backlog items that are small enough to fit within a single iteration, larger backlog items may be split into more fine-grained ones (stories). A number of authors have suggested patterns that can be applied for this purpose, ranging from reducing the feature list to narrowing down the business variations or input channels [Leffl2010]. Note that even fine-grained stories should be defined in such a way that they deliver some value for at least one stakeholder.

Smaller backlog items can be grouped into larger blocks or processes (e.g., use cases) for clarity and displayed graphically using a story map. This helps to maintain an overview of the requirements and at the same time to present the medium-term strategy by assigning the requirements to sprints and releases.

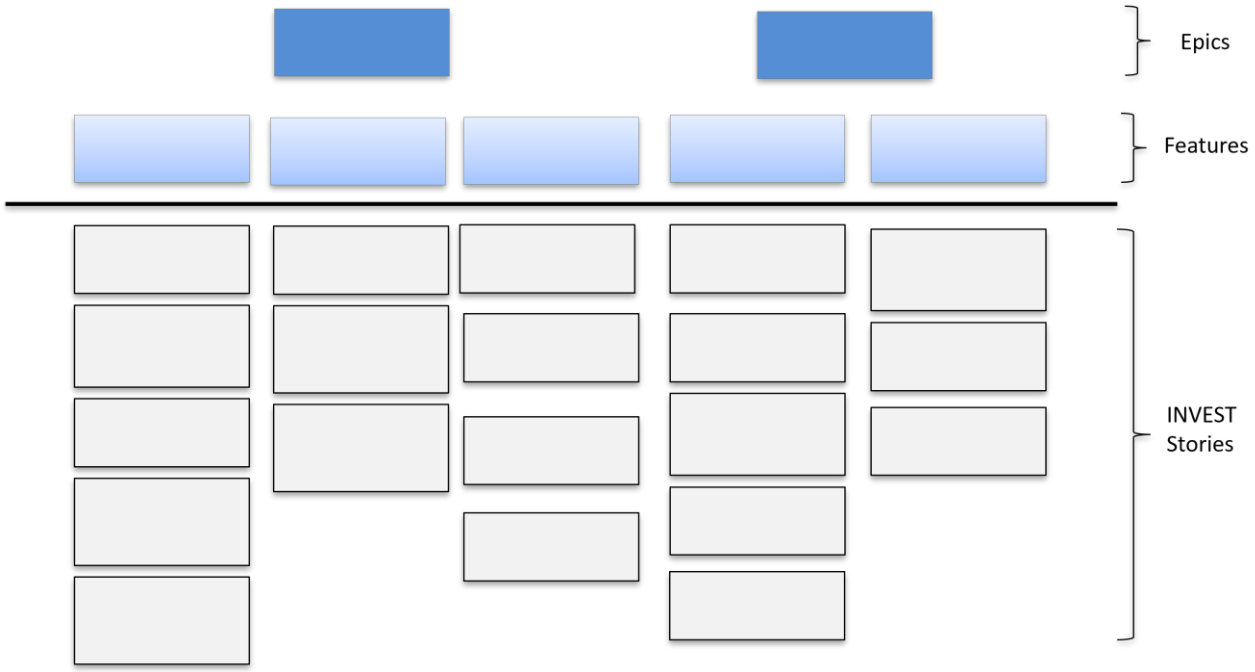

Figure 6: Splitting and grouping

3.5 Knowing when to stop

Educational objectives

EO 3.5.1 Apply the refinement of requirements (L4)
EO 3.5.2 Apply the quality assurance of stories in an agile context (L4)

Content

The product owner is responsible for continuing discussions with developers until both sides have a common understanding of the requirements Meye2014].

For this level of common understanding, the Definition of Ready (DoR) is defined which can be used for the quality assurance of backlog items to ensure that a common understanding and a sufficient level of detail has been achieved.

3.6 Project and product documentation of requirements

Educational objectives

EO 3.6.1 Understand how to distinguish between project and product information/documentation (L2)
EO 3.6.2 Know methods and techniques to preserve information for future use (L1)

Content

A product backlog can be thought of as a replacement for the requirements document of a traditional project. However, it is important to remember that the written part of a backlog item (e.g., using the user story template: "As a user, I want...") is incomplete until discussions about that backlog item have taken place.

It is often best to think of the written part as a reference to a more precise representation of this requirement. Backlog items (Epics, Features, Stories, ...) could point to a diagram depicting a workflow, a spreadsheet illustrating how to perform a calculation, or any other artifact the product owner or team desires.

In addition to the product backlog, there may be different reasons for a detailed requirements documentation, such as for communication purposes, for reflection purposes, for legal purposes, for archiving purposes.

Defining an adequate level of documentation depends on numerous factors, such as the scope of the project, the number of stakeholders involved, legal constraints and/or the safety-critical aspects of the project. Based on these factors, teams in an agile environment try to avoid an excess of documentation and find a minimum denominator of useful documentation.

Working with a "living" product backlog is an efficient way of dealing with documentation, but it is not always sufficient. Structured documentation of all requirements implemented in a product that is up to date is not only legally mandatory in some projects, but it also serves as an ideal starting point for identifying change requests more quickly based on the existing documentation.

4 Handling quality requirements and constraints (L3)

Duration: 90 minutes + 30 minutes exercise

Terms: Quality requirements, Constraints, Quality tree, Definition of Done (DoD), Acceptance criteria

Content

This EU takes a look at quality requirements and constraints in agile projects. Even though the term "non-functional requirements" is still often used in practice as an umbrella term, we use the more concrete and precise categories "quality requirements" and "constraints" according to [Glin2024].

4.1 Understanding the importance of quality requirements and constraints (L2)

Educational objectives

EO 4.1.1 Understand the importance of quality requirements in an agile context (L2)
EO 4.1.2 Understand categorization schemes for quality requirements and constraints (L2)

Content

Many agile methods concentrate on functional requirements only and do not put enough emphasis on qualities and constraints Meye2014].

Key constraints and qualities envisaged for the system should be made explicit early in the lifecycle of a product, since they determine key architectural choices (infrastructure, software architecture and software design). Ignoring them or learning too late in the project may endanger the whole development effort. Other qualities can be captured iteratively, just in time, as with functional requirements Meye2014].

Categorization schemata for quality requirements and constraints (e.g., [RoRo2013], [ISO25010]) can be used as checklists so as not to forget important categories.

4.2 Adding precision to quality requirements (L2)

Educational objectives

EO 4.2.1 Understand the detailing or decomposing of quality requirements and constraints (L2)
EO 4.2.2 Understand the derivation of functional requirements from quality requirements (L2)
EO 4.2.3 Understand the specification of acceptance criteria for quality requirements (L2)
EO 4.2.4 Understand the added value of quality trees (L2)

Content

Initially quality requirements are often deliberately vague. They have to be captured in their vague format as a starting point. Vague quality requirements and constraints can be refined

into more precise requirements. Sometimes concrete functional requirements will be derived from them.

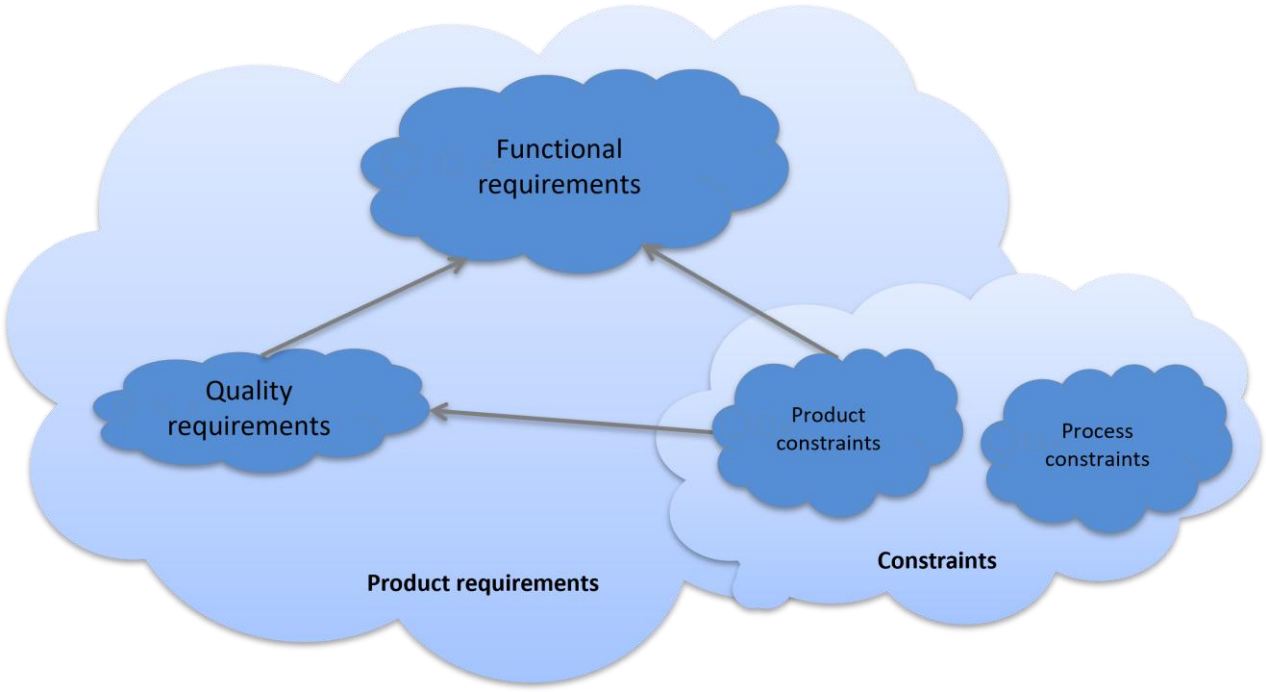

Figure 7: Decomposing quality requirements and constraints

Decomposing (or detailing) a quality requirement means specifying quality requirements on a lower level of detail, e.g. by using the generalizations in the categorization schemes like "usability" and making them more precise by finding requirements for "ease of use" and "ease of learning".

Deriving means that quality requirements can be achieved by defining functional requirements, i.e. suggesting functions that achieve the desired quality or constraints. An example for refining a security requirement is introducing a role concept and passwords.

Quality trees ([BOSS2022], [CleA2001]) are also a proven way to structure quality requirements.

Acceptance criteria must also be defined for quality requirements in order to make them testable at a later stage, just like other types of requirements [PoRu2021]. The type of acceptance criteria used will depend on the category of the quality.

4.3 Quality requirements and the backlog (L3)

Educational objectives

EO 4.3.1 Apply attaching of quality requirements to functional requirements (L3)
EO 4.3.2 Apply creating separate backlog items for quality requirements (L3)
EO 4.3.3 Understand quality requirements as part of the DoD (L2)
EO 4.3.4 Understand the difference between quality requirements and acceptance criteria (K2)

Content

Generalized quality requirements need to be linked to more specific functional requirements [PoRu2021], e.g., some quantifiable throughput attached to a story, or specific security features attached to an epic.

Other qualities, e.g., scalability, maintainability, or robustness should be made known to development and checked in each iteration. A common way of achieving that is including them in the Definition of Done. This is often supported by automated testing [Leffl2010].

Another approach is to have a separate recording (away from the product backlog) of such qualities to keep them visible for the teams e.g., as a common list or in the form of checklists. These requirements are all of equal importance (as they all have to be fulfilled) [Leffl2010].

It is also good practice to make the relationships of functional vs. affected quality requirements visible by setting up a matrix on a wall, indicating the "affected by" relationship with marks in the respective cells.

When structuring a backlog, product owners are often faced with the question of whether a recognized/raised quality requirement is really a quality requirement, an acceptance criteria, or perhaps also an acceptance criteria for a quality requirement.

- Quality requirements refer to quality concerns that are not covered by functional requirements. Such as performance, availability, maintainability, security or reliability.
- Acceptance criteria are criteria that a requirement (this can be a functional requirement or a quality requirement) must fulfill in order to be accepted by the stakeholders.

We can see that both functional requirements and quality requirements can and should have acceptance criteria.

4.4 Making constraints explicit (L2)

Educational objectives

EO 4.4.1 Understand different kinds of constraints in an agile context (L2)
EO 4.4.2 Know how to characterize constraints (L1)

Content

Constraints are an important type of requirements that limit the design choices of the developers [Glin2024]. Constraints can be categorized as either product constraints or process constraints. Product constraints include the required use of technologies, the reuse of existing components, make or buy decisions, or resources in the form of material, knowledge and competencies. The process constraints, on the other hand, are defined by organizational or development processes. These include organizational policies and regulations, financial limits, norms and standards, compliance regulations and audits, legal and governmental constraints.

It is important to make such constraints explicit so that everyone in the team is aware of them. The most limiting ones should be known early in the project. Others should be captured as soon as they are discovered. In general, constraints usually affect several functional

requirements. This raises the question of how constraints (as well as quality requirements) should be documented. We have already explained the possibilities for this in chapter 4.3. The same applies to the constraints: at the very least, a check should be included in the Definition of Done to ensure that they have been met.

Please note that you do not necessarily have to write all constraints as a backlog item. It may be sufficient to inform the team that e.g., C# and ORACLE are non-negotiable constraints.

Such constraints are normally applicable to a wider range of projects. Once they have been recorded, they can easily be reused in other projects or products.

5 Prioritizing and estimating requirements (L3)

Duration: 120 minutes + 90 minutes exercise

Terms: Business value, MVP, MMP, Planning poker, Cone of uncertainty, Velocity, Sizing, Reference Stories, T-shirt sizes, Fibonacci sequence

Content

Even in a perfect agile world forecasts are needed to determine how much work can be "done" within a previously specified iteration (timebox). Additionally, development organizations that exceed one team usually need forecasts in order to prioritize and plan work properly.

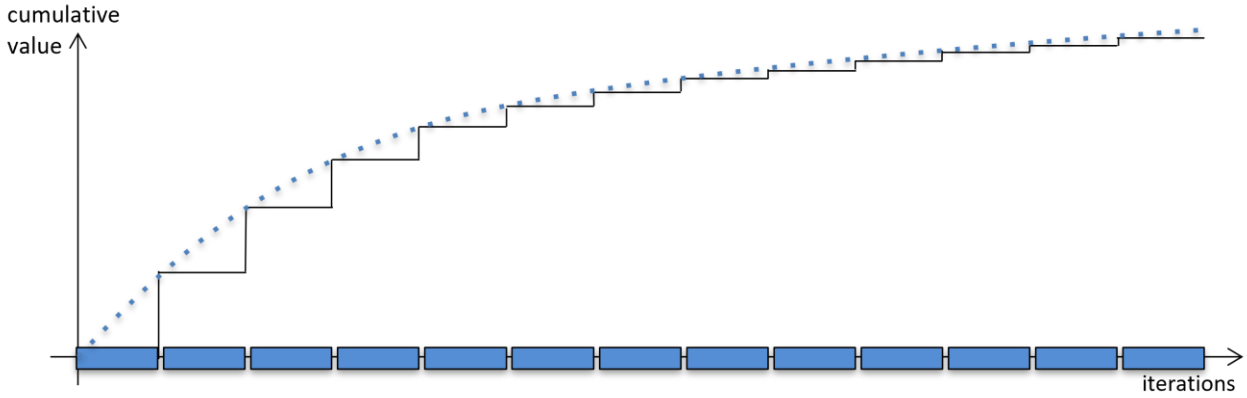

Figure 8: Growing business value

5.1 Determination of business value

Educational objectives

EO 5.1.1 Understand the determination of business value (L2)
EO 5.1.2 Understand how to use business value to order backlog items (L2)
EO 5.1.3 Apply alternative methods of calculating the business value (L3)
EO 5.1.4 Understand how to align business value measurement to strategic goals of the organization (L2)

Content

Agile approaches aim to maximize the overall business value and to permanently optimize the overall business value creation process [Leffl2010]. In a product backlog, all requirements (whether coarse or fine) should be ordered primarily by the value they provide to the organization. A prerequisite to doing so is an agreed definition of what business value for this product/company is.

Business value is not only defined by profit: alternative calculations include Return on Investment, Payback Period, Net Present Value, Weighted Shortest Job First (WSJF), Cost of Delay and Balanced Scorecard. Market value, time to market and reducing potential risks

all potentially represent types of business value, as do operational and organizational excellence [Rein2009].

Indeed, the definition of business value may be different in every organization, every project, and from the perspective of different stakeholders. Professionals should understand how to align business value measurement to the strategic goals of the organization, and be able to adapt this alignment as these goals change.

5.2 Business value, risks, and dependencies (L3)

Educational objectives

EO 5.2.1 Understand the dependencies between potential business value and related risks (L2)

Content

Very often potential business value and risks are interdependent. Focusing on a specific business value might raise specific risks, changing the focus of the business value might change the risks as well [Rein2009].

In each case the ordering of requirements should be adjusted in line with the selected strategy, taking into account dependencies among the requirements.

5.3 Expressing priorities and ordering the backlog

Educational objectives

EO 5.3.1 Apply the prioritization of backlogs (L3)
EO 5.3.2 Apply different basic prioritization strategies (L3)
EO 5.3.3 Apply the determination of dependencies between requirements (L3)
EO 5.3.4 Understand the sequence of backlog items based on their dependencies (L2)

Content

Once you have determined what value means to you, you have to express these priorities and order the backlog according to the priorities given to the backlog items. There are many different methods to assign value to backlog items. Some of them very simple, others are highly complex. Examples of prioritization strategies include:

- MoSCoW
- High/Medium/Low
- Based on a range of numbers that express the business value
- Based on valuing the business using multiple criteria

Backlog Items	Turnover in Q2		Minimizing technical risk		Increasing usability		Overall result
	Weight = 5	Value	Weight= 4	Risk value	Weight= 2	Usability Value	
Requirement 1	3	15	0	0	0	0	**15**
Requirement 2	0	0	3	12	1	2	**14**
Requirement 3	4	20	2	8	2	4	**32**
Requirement 4	2	10	2	8	3	6	**24**
Requirement 5	0	0	2	8	5	10	**18**
...							
...							

Figure 9: Priorities and ordering the backlog

When prioritizing the backlog items, it should generally be considered that the closer the time of the planned implementation of a backlog item comes, the clearer the priorities of the selected items should be. A prioritization with the criteria "high", "medium", "low" often results in a far too large number of backlog items receiving the value "high". For example, if 30% of all backlog items are assigned a high priority, the result is that developers do not know what is most important to the product owner. It also indicates that the product owner does not have a clear strategy for short to medium-term implementation. The goal of prioritization should always be to make a clear statement about what stakeholders can expect as the value of the product in the near future.

5.4 Estimating backlog items (L3)

Educational objectives

EO 5.4.1 Apply forecasts and estimates (L3)
EO 5.4.2 Understand how to derive a mid-term forecast (L2)
EO 5.4.3 Understand the advantage of relative, categorizing and group estimations (L2)
EO 5.4.4 Understand estimation techniques (L2)

Content

Initial project estimates are often imprecise. They become more and more precise as the activity is iterated (a principle known as the Cone of Uncertainty). By analyzing what has been delivered in previous iterations, the velocity of the team can be calculated. This allows the capacity for future iterations to be better estimated.

For better mid-term estimates, large requirements such as epics are broken down into features. This allows estimates to be made at feature level and then added up to the epic. Although these estimates are still not very precise, but helpful for the epic estimation. In addition, they serve as a kind of test of knowledge about the Epic (assumptions, etc.).

Agile methods define rules that help to do better and more accurate estimates:

- Everyone involved in the estimation must have the same understanding of the work that needs to be "done".
- Estimations must be performed by those doing the work, the cross-functional team (Developers in Scrum). This helps to bring all involved people on the same level of knowledge by exchanging knowledge and assumptions about the work to be done.
- Estimations should be done relatively / relative to work already done (Estimation by analogy), since those estimates are more likely to be accurate than absolute estimates.
- Estimates should be done in an artificial unit representing effort, complexity and risk in one.

Several techniques support the relative estimate. The best known of these are planning poker [Cohn2006] and Magic Stimulation.

5.5 Choosing a development strategy (L2)

Educational objectives

EO 5.5.1 Understand the concept of MVP (minimum viable product) (L2)
EO 5.5.2 Understand the concept of MMP (minimum marketable product) (L2)

Content

Different strategies can be applied when selecting what should be picked for early releases, based on known value, risk and effort needed to develop a backlog item. Two concepts are typical for agile development: developing a minimum viable product (MVP) and developing a minimum marketable product (MMP).

A minimum viable product is the version of a new product that allows a team to collect the maximum amount of validated learning about customers with the least effort. The MVP is the central idea of the Lean Startup methodology developed by Eric Ries, which is based on the Build–Measure–Learn cycle. The MVP is not necessarily a deployable software product.

The MMP describes the product with the smallest possible feature set that meets the needs of the first users (innovators and early adopters) and can therefore be marketed.

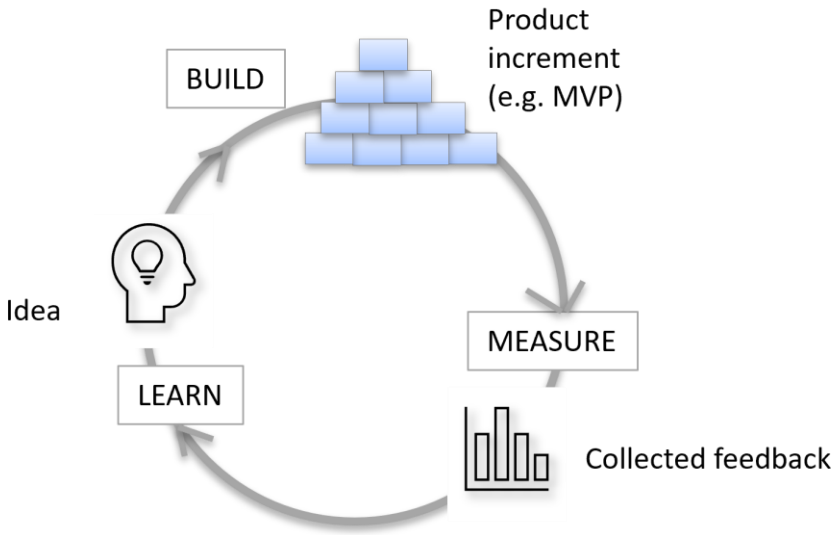

Figure 10: The “Build-Measure-Learn” cycle of lean development

6 Scaling RE@Agile (L2)

Duration: 105 minutes

Terms: Scaling frameworks, Scaling Requirements and Teams, Structuring Requirements and Teams in the Large, Roadmaps and Large Scale Planning, Product Validation

Content

RE is easier for products that are small enough to be handled by a single team at one location.

In this educational unit we discuss why product development must sometimes be scaled and why products have to be developed by more than one team, whether at the same location or distributed geographically. When scaling, the product owner of the overall product (as the role responsible for requirements management) is likely to be more challenged with management aspects than with requirements aspects. We will discuss that the two factors time to market and complexity (either functional complexity or challenging quality requirements) justify and drive the scaling process. But organizational and technical constraints also influence the way scaling takes place.

The following questions are important:

- What does scaling mean and how does it affect requirements and teams?
- How do we (re-)organize the requirements and the teams in the large?
- How are releases and roadmaps defined and used in long-term planning?
- How are requirements validated in scaled environments?

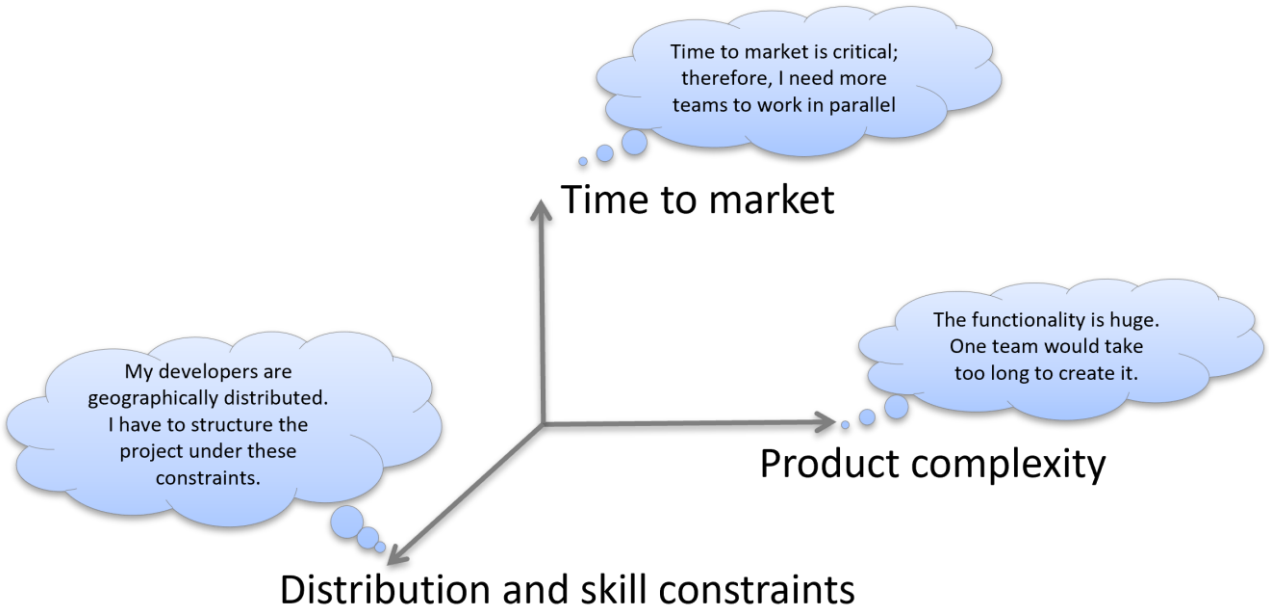

Figure 11: Reasons to scale

6.1 Scaling requirements and teams (L2)

Educational objectives

EO 6.1.1 Know common examples of scaling frameworks (L1)
EO 6.1.2 Understand the challenges and mechanism for scaling requirements in the agile context (L2)

Content

We use the term "scaling" to describe a change in size, either of the system or the product, or of the number of people involved.

Since around 2010, a number of different agile scaling frameworks have been developed to address these issues. Among them are Nexus [Nexu2021], SAFe [SAFe2021a] [SAFe2021b], LeSS [LeSS], Scrum@Scale [S@SG2021], BOSSA Nova [BOSS2022], Scrum of Scrums [KnIvS2012], and Spotify [Spot2012], though more exist. Scaling frameworks vary in their maturity level, the number of good practices, guidelines and rules, and the degree of adaptability to the specific needs of an organization.

When you scale, two things will always be true: you will be forced to add a hierarchy to the requirements and a hierarchy to the organization. Coarse-grained requirements are needed when discussing the product as a whole; fine-grained requirements will be needed in the teams implementing some aspect of the product. And the teams themselves will need to organize their cooperation to function successfully within a larger team.

6.2 Criteria for structuring requirements and teams in the large (L2)

Educational objectives

EO 6.2.1 Know the challenges to organize a backlog and to communicate about requirements within a network of teams (L1)
EO 6.2.2 Understand requirements splitting in different (project) settings of the agile context (L2)

Content

In large-scale product development mostly multiple teams have to work together on the same product. In practice, each team develops a specific part of the product that must be integrated with other parts into the overall product in order to create a working solution. Only the integrated product has value for the stakeholders.

When scaling product development to multiple teams, it is not sufficient for all product owners to simply meet and somehow discuss which teams should develop which part of the product, and then to hope for the best! Sophisticated structures and practices are needed to support team collaboration, manage requirements changes and enable rapid product delivery. Otherwise, developers may waste effort coordinating with teams that are not relevant for their work.

From a requirements perspective we have to close the loop: from the initial (business-) requirement demanded by stakeholders, through the splitting of complex requirements into smaller pieces manageable by developers, and then onto ensuring that the assembled results combine to form a solution that can be released to the business.

In order to both work on requirements collaboratively, and to take reasonable decisions autonomously, teams need a general understanding of the requirements of the other teams with whom they have to collaborate, without, though, becoming overwhelmed with all the details. Product owners should therefore find an appropriate level of detail, sufficient for teams to understand the impact of their decisions on other teams.

To deliver shippable product increments with minimal dependencies on other teams, teams in an agile environment should work on loosely-coupled, end-to-end features. In our context, the term 'end-to-end feature' refers to a set of coherent functions performing a specific task that provides business value to stakeholders. Use Cases are an approach to structuring requirements, not always typically associated with Agile, but nevertheless recommended by a number of authors (for example [Jaco2011], Cockburn, [Leffl2010]).

Unfortunately, in many cases it is not that easy to decompose requirements based around loosely-coupled units of end-to-end functionality. Due to architectural design (for example technology, infrastructure, system components, common platform, architectural layers such as front- and backend) as well as organizational considerations (specialist skills, team location, sub-contractors), units of functionality may overlap. This means that different teams in an agile environment must work together to implement specific features and their respective product owners need to collaborate more closely on requirements. Alternatively, a dedicated team can be established to specifically work on the overlap, and to collaborate with each of the original teams focused on a unit of functionality.

6.3 Roadmaps and large scale planning (L2)

Educational objectives

EO 6.3.1 Know the difference between a roadmap and a backlog in an agile context (L1)
EO 6.3.2 Understand the creation and the management of a roadmap in the agile context (L2)

Content

In large-scale product development, product owners manage requirements in the product-focused backlog. In contrast to the backlog, a roadmap is used for planning product development incrementally. A roadmap is a prediction of how the product will grow [Pich2016]. Roadmaps do not change the content of backlog items but arrange them onto a timeline. It answers the question when we can roughly expect which features.

A roadmap is a useful means to communicate (strategic) goals and decisions to the developers and other stakeholders. It breaks down a long-term goal into manageable iterations, represents dependencies among the teams and provides direction and transparency to the stakeholders.

A roadmap shows strategic goals, milestones and coarse-grained requirements. Important milestones may be either internal or determined by external events such as a trade show or the introduction of new regulation to the market. The representation of a roadmap depends on its purpose, target group and planning horizon.

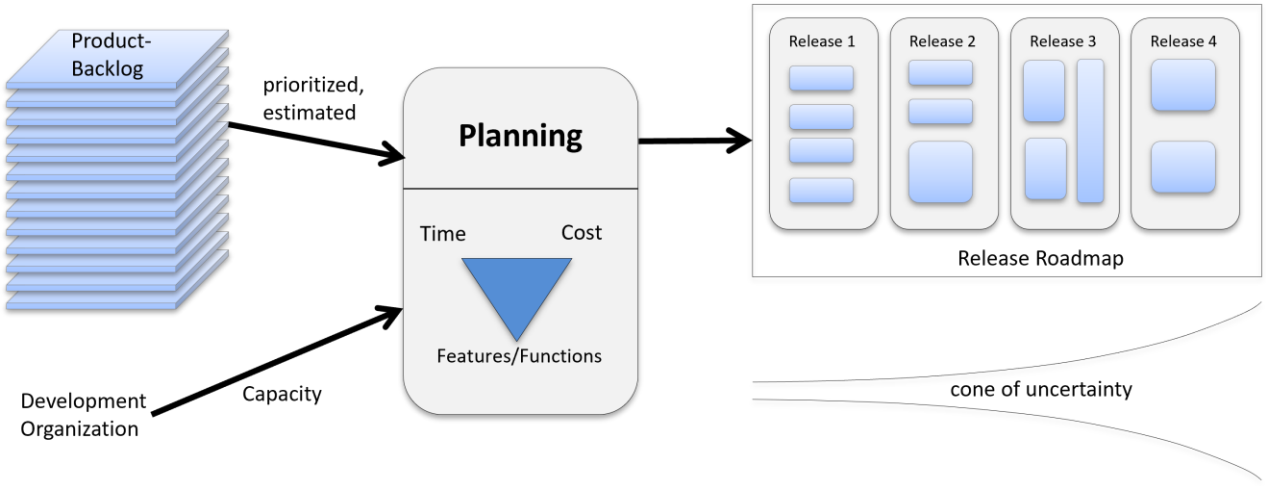

Figure 12: Roadmaps and large scale planning

To develop a long-term product roadmap, a product owner must first define a product vision and strategy. Afterwards, product owners must then elicit coarse-grained requirements by engaging with the necessary stakeholders. There is no need to invest time on detailed requirements at this point. Although requirements are subject to a high level of uncertainty at this stage, the product roadmap as a rough, first-cut iteration plan is good enough to support planning and synchronization.

6.4 Product validation (L2)

Educational objectives

EO 6.4.1 Understand concrete methods to validate product requirements in the agile context (L2)

Content

A key idea of agile development is to develop a small slice of the product, generate feedback by involving stakeholders and adapt the product development according to the findings and insights gained. Thus, following the principle of the Build-Measure-Learn cycle [Ries2011], product validation becomes an important step to gain rapid feedback. Each time a new product increment is released, product owners use that product increment to verify its business value and to examine whether the product requirements had been correctly understood.

Validation at product level (e.g., sprint review in Scrum) is an important method in large-scale product development, as it ensures that the product owners together share full accountability from business requirements to product integration. It is the whole product that has value for the stakeholders, not only small product slices.

Another approach for product validation in large-scale product development is one that is based on data analysis [MaeA2016]. The integrated product increment is delivered to users and, based on their behavior, measurements are made as to whether the product features have a positive, neutral or negative impact.

7 Definitions of terms, glossary

For the definitions of terms, we refer the reader to the IREB CPRE Requirements Engineering glossary [Glin2024], which is not only a comprehensive glossary of Requirements Engineering terminology, but also defines many terms from the field of agility. For specific agility terms, the reader may consult the current Scrum Guide [ScSu2020].

8 References

[Alex2005] Alexander, I. F.: A Taxonomy of Stakeholders – Human Roles in System Development. International Journal of Technology and Human Interaction, Vol 1, 1, 2005, pages 23–59.

[BOSS2022] https://www.agilebossanova.com/#bossanova. Last visited March 2025.

[CleA2001] P. Clements et al.: Evaluating Software Architectures, SEI Series in Software Engineering, 2001

[Cohn2010] Cohn, M.: User Storys für die agile Software-Entwicklung mit Scrum, XP u.a., mitp, 2010

[Cohn2006] Cohn, M.: Agile Estimation and Planning, Addison Wesley, 2006

[Dora1981] Doran, G. T: There's a S.M.A.R.T. way to write management's goals and objectives, Management Review. AMA FORUM. 70 (11): 35–36 1981.

[Glin2024] Glinz, M.: A Glossary of Requirements Engineering Terminology. Standard Glossary for the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version 2.1.1, 2024. https://cpre.ireb.org/en/knowledge-and-resources/downloads#cpre-glossary. Last visited March 2025.

[HeHe2011] Heath, C., Heath, D.: Switch: Veränderungen wagen und dadurch gewinnen. Crown Business, 2010

[High2001] Highsmith, J.: Design the Box. Agile Project Management E-Mail Advisor 2001, http://www.joelonsoftware.com/articles/JimHighsmithonProductVisi.html. Last visited March 2025.

[Hrus2017] https://b-agile.de/downloads/articles/story_splitting.pdf. Last visited March 2025.

[ISO25010] ISO/IEC 25010:2023: Systems and software engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and software quality models: https://www.iso.org/standard/78176.html. Last visited March 2025.

[Jaco2011] https://www.ivarjacobson.com/publications/white-papers/use-case-ebook. Last visited March 2025.

[Leffl2010] Leffingwell, D.: Agile Software Requirements – Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison Wesley, 2010.

[LeSS] Large-Scale Scrum: https://less.works. Last visited March 2025.

[MaeA2016] Maalej, W., Nayebi, M., Johann T., Ruhe, G.: Toward Data-Driven Requirements Engineering. IEEE Software (Volume 33, Issue 1), 2016.

Meye2014] Meyer, B.: Agile! The Good, the Hype and the Ugly, Springer, 2014.

[Nexu2021] https://www.Scrum.org/resources/nexus-guide. Last visited March 2025.

[Pich2016] Pichler, R.: Strategize – Product Strategy and Product Roadmap Practices for the Digital Age, Pichler Consulting 2016.

[PoRu2021] Pohl, K., Rupp, C.: Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level, dpunkt.verlag, 5. Auflage, 2021.

[Prim2017] CPRE RE@Agile Primer https://cpre.ireb.org/en/downloads-and-resources/downloads#cpre-agile-primer-syllabus-and-study-guide. Last visited March 2025.

[Rein2009] Reinertsen, D.G.: The Principles of Product Development Flow – Second Generation Lean Product Development. Celeritas Publishing, 2009.

[Ries2011] Ries, E.: The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, New York, NY 2011.

[RoRo2013] Robertson S. Robertson J.: Mastering the Requirements Process – Getting Requirements Right, 3rd edition, Addison Wesley, 2013.

[Robe2003] Robertson, S.: Stakeholders, Goals, Scope: The Foundation for Requirements and Business Models, 2003, https://www.volere.org/wp-content/uploads/2018/12/StkGoalsScope.pdf. Last visited March 2025.

[SAFe2021a] https://www.scaledagileframework.com/roadmap/. Last visited March 2025.

[SAFe2021b] https://www.scaledagileframework.com/pi-planning/. Last visited March 2025.

[SAFe2021] https://www.scaledagileframework.com/safe-requirements-model/. Last visited March 2025.

[S@SG2021] Sutherland, J. and Scrum, Inc: Scrum@Scale Guide: https://www.Scrumatscale.com/Scrum-at-scale-guide/. Last visited March 2025.

[KnIvS2012] Kniberg, H.; Ivarsson, A.: Scaling Agile @ Spotify with Tribes, Sqads, Chapters & Guilds. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Last visited March 2025.

[Spot2012] https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Last visited March 2025.

[ScSu2020] Schwaber, K., Sutherland, J.: The Scrum Guide. 2020 https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf. Last visited March 2025.

[Wake2003] Wake, B: INVEST in Good Stories, and SMART Tasks, 2003, https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Last visited March 2025.