Agile Planning

Dirk Riehle, Univ. Erlangen

AMOS B04

Final Reminder About Common Mistakes

Please sign-off your commits and declare your co-authors, if any

During sprint review, please show and tell, not just tell

Don’t forget your sprint preparation meeting

Agenda

1. Product goal
2. Product glossary
3. Product backlog
4. Sprint planning
5. Release planning
6. Definition of done
7. Roadmapping

1. Product Goal

Product Goal [1]

The **product goal** is

- The purpose of undertaking the project

To resolve the product / project conflict, AMOS separately defines

- Product vision and project mission

[1] Introduced only in the Scrum 2020 guide

Product Vision

The **product vision** is the

- Timeless reason why the software under development should exist

The product vision should contain a sustainability model

- Business value of why someone pays for the development

The Flowers social network helps flower enthusiasts worldwide to connect with each other and enjoy following their favorite hobby online. Centered on showing and rating favorite flower photos, it inspires growing and presenting ever more beautiful flowers. With a highly engaged user community, Flowers is the best place for producers and sellers of gardening supply to reach out to customers and engage with them. Such engagement involves understanding flower enthusiasts' needs around gardening supplies and selling to them.

Project Mission

The **project mission** is

- What the team has committed to achieving within the given project time-frame

The mission of this project is to create an MVP for Wahlzeit with the Flowers extension. Core functionality will be showing and rating photos, basic user management, case management, and minimal system administration.

One-Time Deliverable: Product Vision and Project Mission

Please define and agree on product vision and project mission

If necessary, update vision and mission during the project

Product Goal / Product Glossary / Product Backlog

2. Product Glossary

Product Glossary

A **product glossary** is a

- List of domain concept (term) definitions

Domain concepts can be

- Original concepts, synonyms (links), shorthands, …

A glossary is a poor man’s approach to a domain model

- Lack of formality doesn’t necessarily make it easier

The domain is the **application domain**

Example Domain Glossary

Term	Definition
Photo	A photo is an image uploaded by a user for display as part of the user's photo portfolio
Photo rating	A short-hand for either individual or community photo rating
Individual photo rating	An integer value of 1..10 that a user gives to a photo shown to them
Community photo rating	A rational value of 1..10 that is the average of all individual photo ratings
Photo status	The status of a photo within the Wahlzeit system (uploaded, published, etc.)

Common Mistakes and Best Practices

Common mistakes

- Lack of precision
- Confusing application with technical domain
- Redundant definitions

Best practices

- Work from first principles i.e. “is a” (supertypes)
- Avoid redundancy by building terms on each other
- Make terms mutually exclusive, completely exhaustive

Regular Deliverable: Product Glossary

Please create a product glossary and keep it up-to-date

3. Product Backlog

Scrum Backlogs

A **backlog** is a

- Prioritized list of items that need doing

The **product backlog** is a backlog of items that

- Are expected of the software under development

The **sprint backlog** is a backlog of items that

- Are marked for doing in the upcoming sprint

The **impediments backlog [1]** is a backlog of items that

- Are impediments to resolve and improvements to make

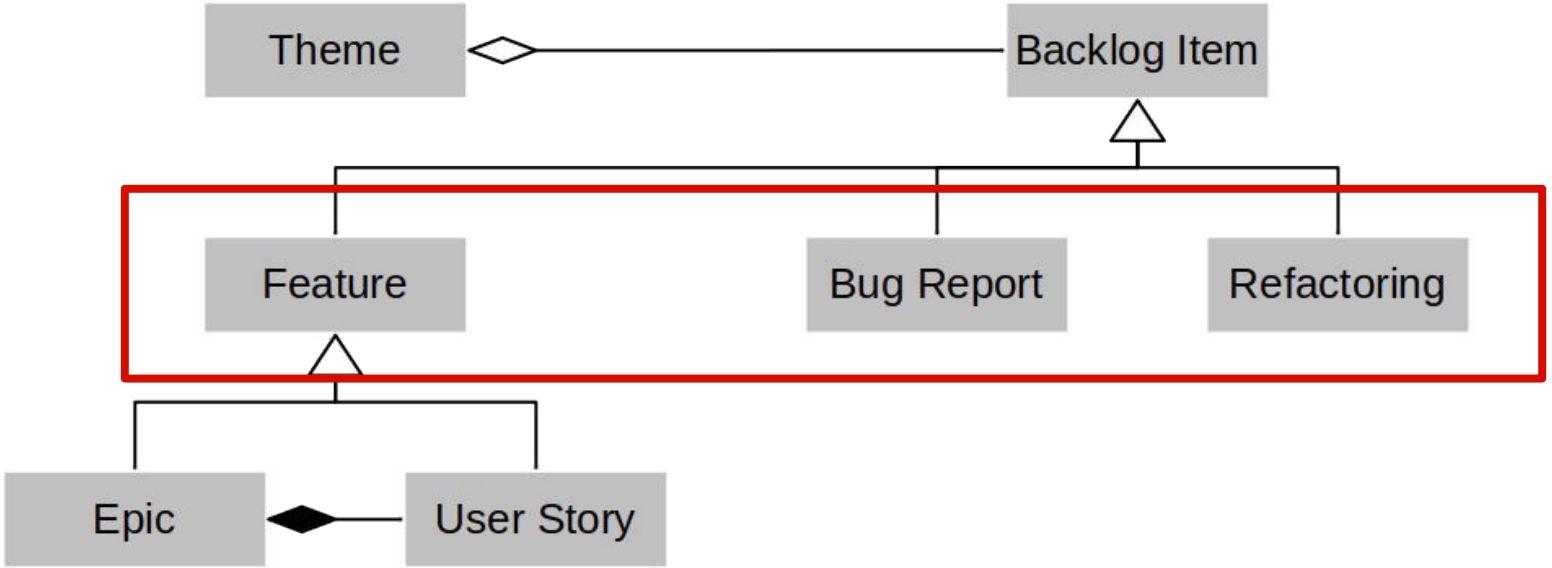
[1] Called imp-squared (impediments x improvements) backlog in AMOS

Backlogs and Backlog Items

Backlog items are items in a backlog

- Product backlog → product backlog items
- Sprint backlog → sprint backlog items
- Impediments backlog → impediments x improvements

Types of Backlog Items [1] [2]



[1] If you are using Jira, backlog items are also called “tickets”
[2] There can be more types of backlog items, for example, “spikes”

Features, Refactorings, and Bug Fix Requests

A **feature** is

- A distinguishing characteristic of a software item [IEEE 829]

A **refactoring** is

- A behavior-preserving code transformation to improve code quality

A **bug fix** is

- A bug report where the bug is to be fixed against the underlying feature

Epics and User Stories

An **epic** is

- A large feature awaiting break-down into smaller features
- A placeholder for these smaller features

A **user story** is

- A feature presented using a the user-story-pattern that is
- Small enough to be implemented in a sprint

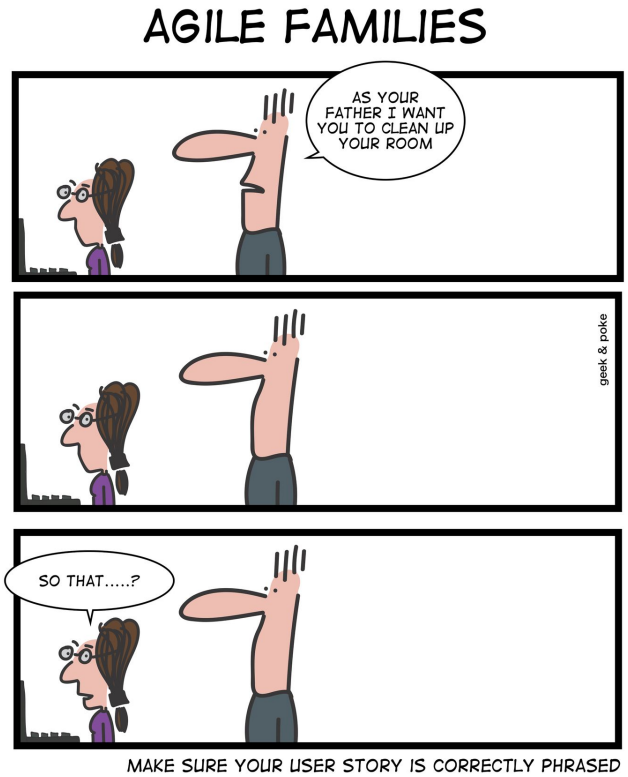
User Stories

A user story is a feature described using a pattern of

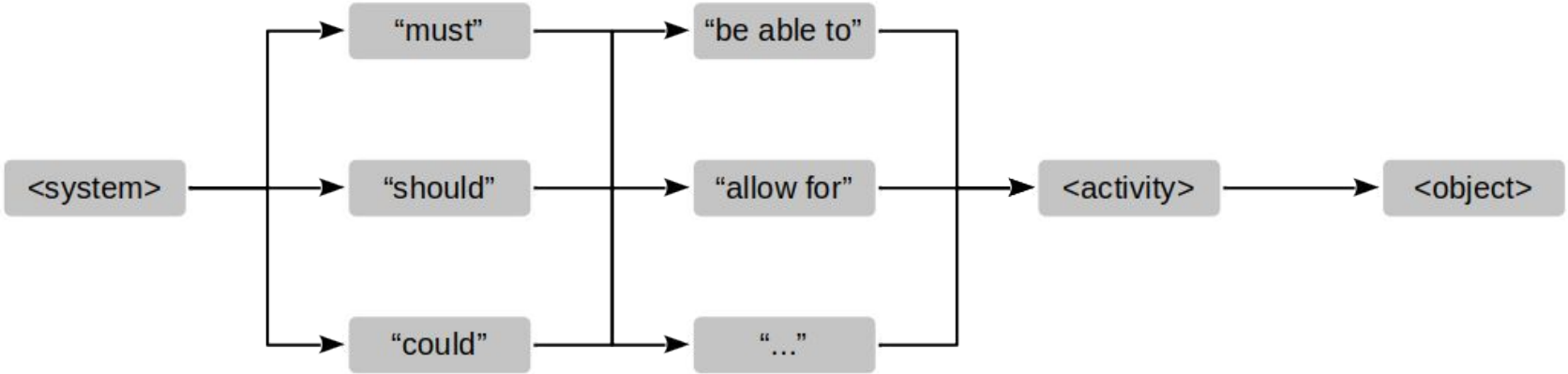
- As a **[user role]**
- I need a **[function]** so that
- I get **[business value]**

User stories are discussion starters, not specifications

Tell-a-Friend: As a **Flowers user,** I need a function to **tell a friend about a flower photo,** so that I can **share my passion for flowers and increase my network.**



Common Alternative to User Stories



Quality Criteria for Backlog Items

I	ndependent: Items should be independent of each other
N	egotiable: An item can be questioned and revised
V	aluable: An item should have recognizable business value
E	stimatable: An item should be sufficiently precise to estimate a size
S	mall: An item should be small enough to fit into one iteration
T	estable: An item should have testable success criteria

Acceptance Criteria

An **acceptance criterion** for a backlog item is

- A proposition that must be true before the item can be accepted

Acceptance criteria are the list of required propositions

- Acceptance criteria are specific to the backlog item

Acceptance criteria are written by the product owner

Story Points

Story points

- Is an arbitrary numeric measure of size of a given backlog item

Properties

- Is a measure of size, not of effort or duration
- Measured in non-linear increments, forcing choice
- Is socially agreed upon, depends on team estimation history
- Is independent of a particular person (and their skills)
- Is mapped to time using the team's velocity (development speed)

Points	Meaning
0	No size
1	Trivial size
2	Small size
3	Medium size
5	Large size
8	Very large size
13	Too large (size)

Size vs. Effort

Size is

- An estimate of complexity
- Measured in an arbitrary unit
- Does not depend on people

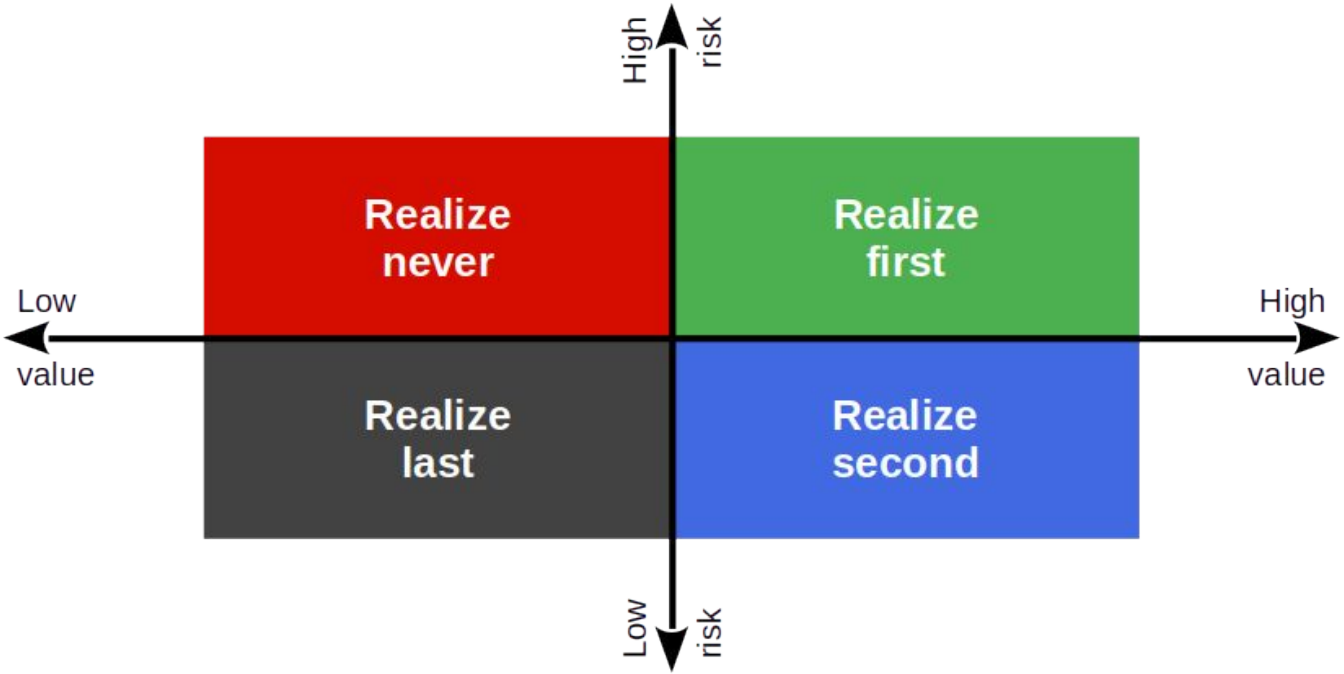
Effort is

- An estimate of duration
- Measured in person hours
- Depends on the implementer

Prioritization by Dependency

A backlog items precedes their dependent backlog items

Prioritization by Risk / Reward



Product Backlog Items vs. Tasks

Product backlog items are

- Written by a product owner
- Business-value-oriented

Tasks are

- Written by a software developer
- Implementation-oriented

Product backlog items can be broken down into tasks

4. Sprint Planning

Sprint Duration

Sprints are

- Same-duration time-boxes that deliver a useful increment of value

Realistic durations in practical use

- One-week durations (like AMOS, but less common)
- Two-week durations (most common sprint duration)
- One-month durations (in use, but too long for some)
- Six-months durations (not really agile any longer)

Types of Sprints

1. Regular sprints
2. Exploratory sprints
3. Cleanup sprints
4. Release sprints

Sprint Planning

In the run-up to sprint planning, the product owner

- Proposes a sprint goal
- Prepares the product backlog

During sprint planning, the Scrum team

- Agrees on the sprint goal
- Plans as discussed before

Sprint Goal

The sprint goal is

- The purpose of the sprint (crisply formulated)

The product owner proposes it, but

- The developers agree and commit to it

The sprint goal discussion is the first part of sprint planning

Regular Deliverable: Sprint Goal

Please agree on a sprint goal during sprint planning

- Add your sprint goal to your planning document

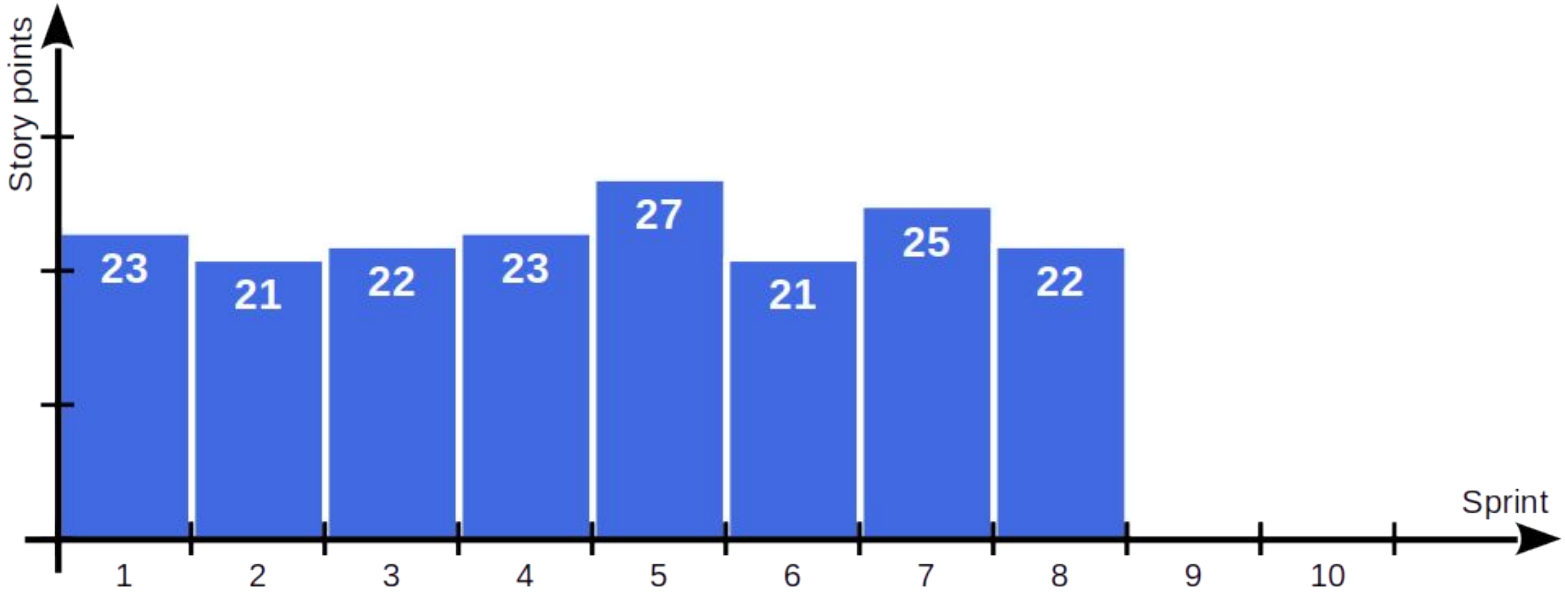
Development Speed (Velocity)

$$v = s / t$$

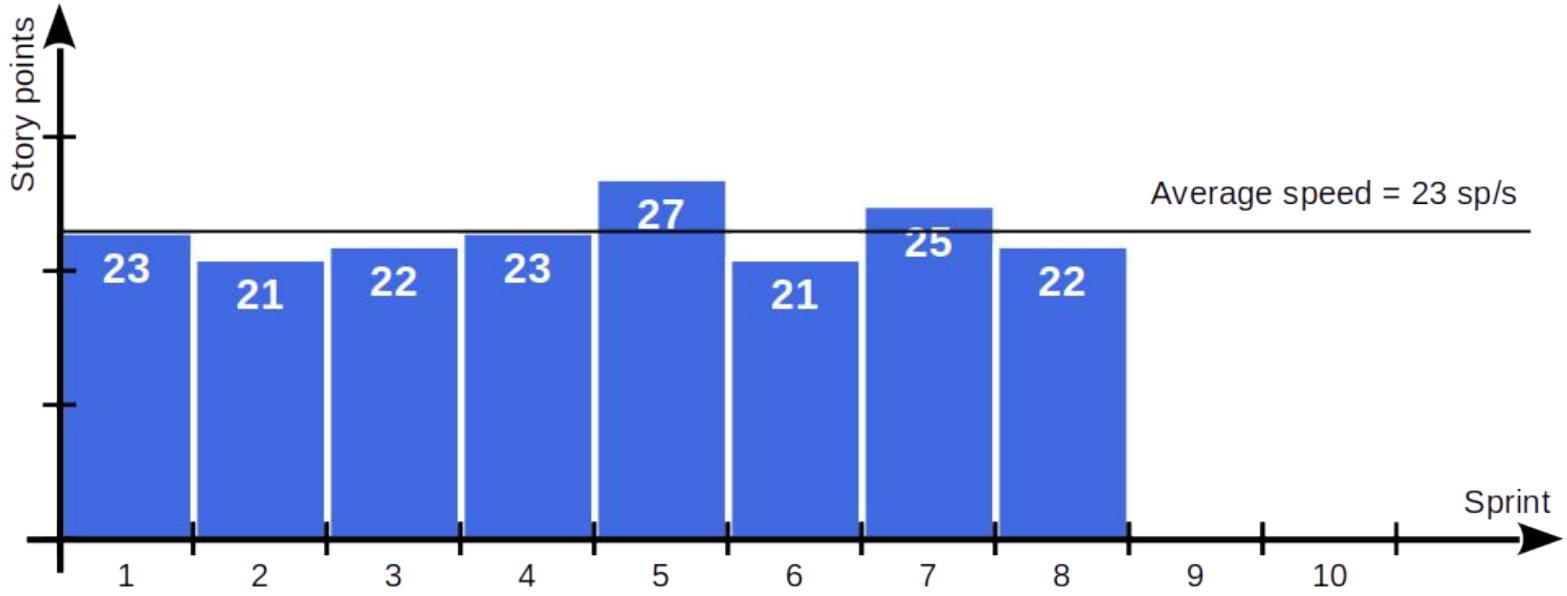
(Story points per sprint)

v = speed (velocity)
s = size (of feature)
t = time (in sprints)

Charting Development Speed

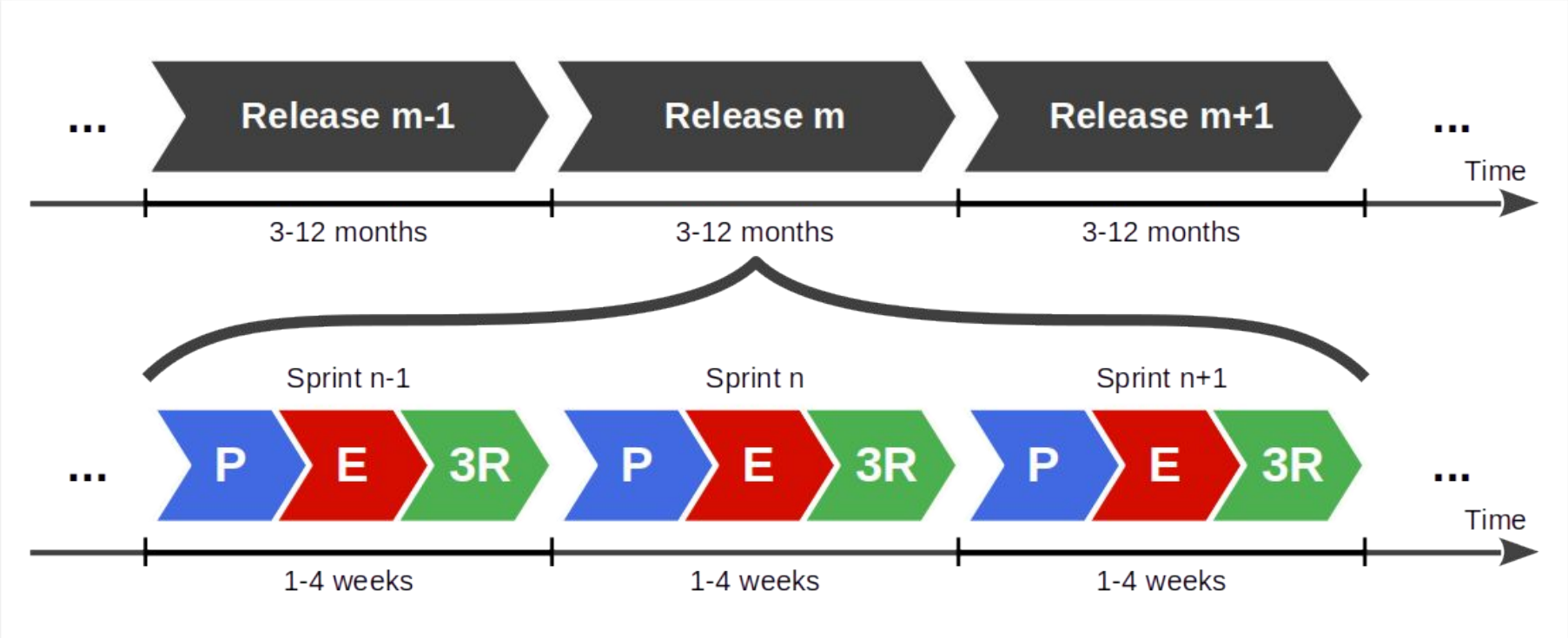


Measuring Average Speed for Sprint Planning



5. Release Planning

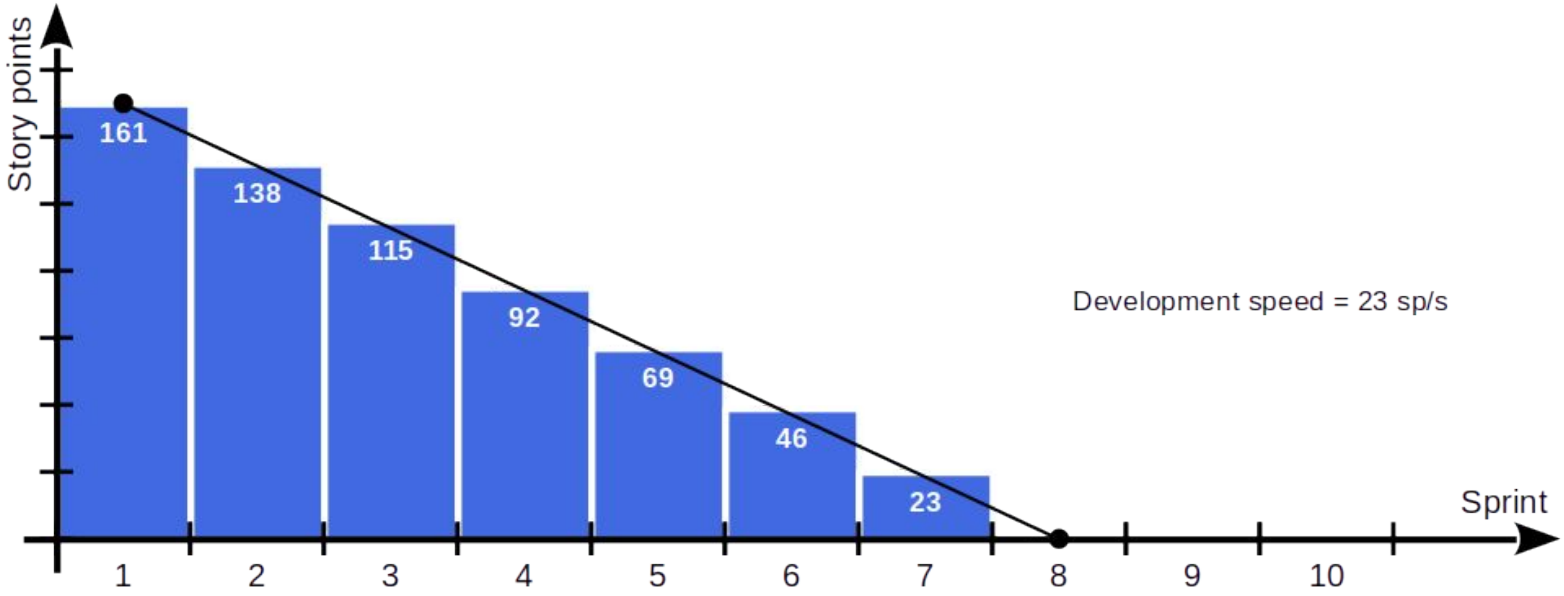
Software Development as a Sequence of Releases



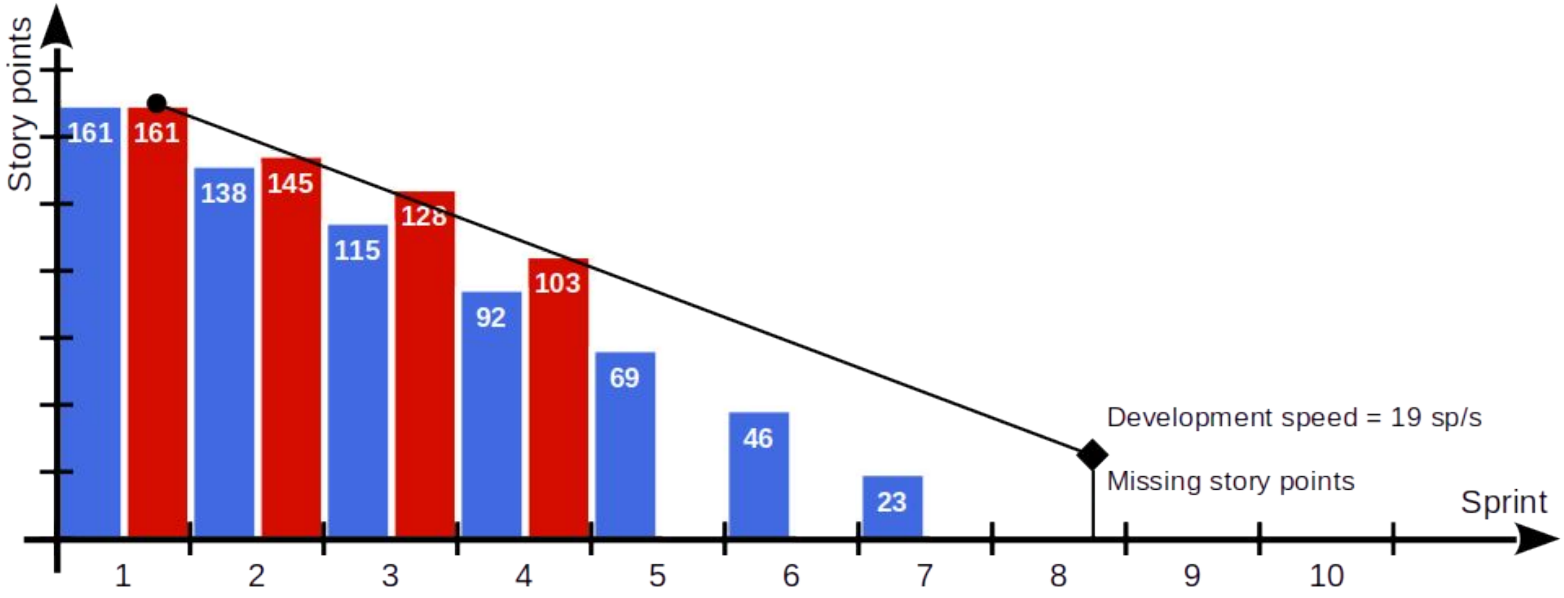
Example of a Project / Product Release Plan

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
1	Deliver first increment of running software		21	63	21	63
2	Deliver increment with basic user handling		21	42	23	42
3	Deliver increment with basic photo handling		21	21	0	19
4				0		19
Features						
1	Deliver first increment of running software					
		Register	8		8	
		Login	5		5	
		Logout	3		3	
		Reset Password	5		5	
2	Deliver increment with basic user handling					
		Prompt Basic Profile	5		5	
		Change Basic Profile	5		5	
		Change Password	3		5	
		Upload Photo	8		8	
3	Deliver increment with basic photo handling					
		Browse Photo Portfolio	8			
		Select Photo	5			
		Change Photo Data	3			
		Delete Photo	5			

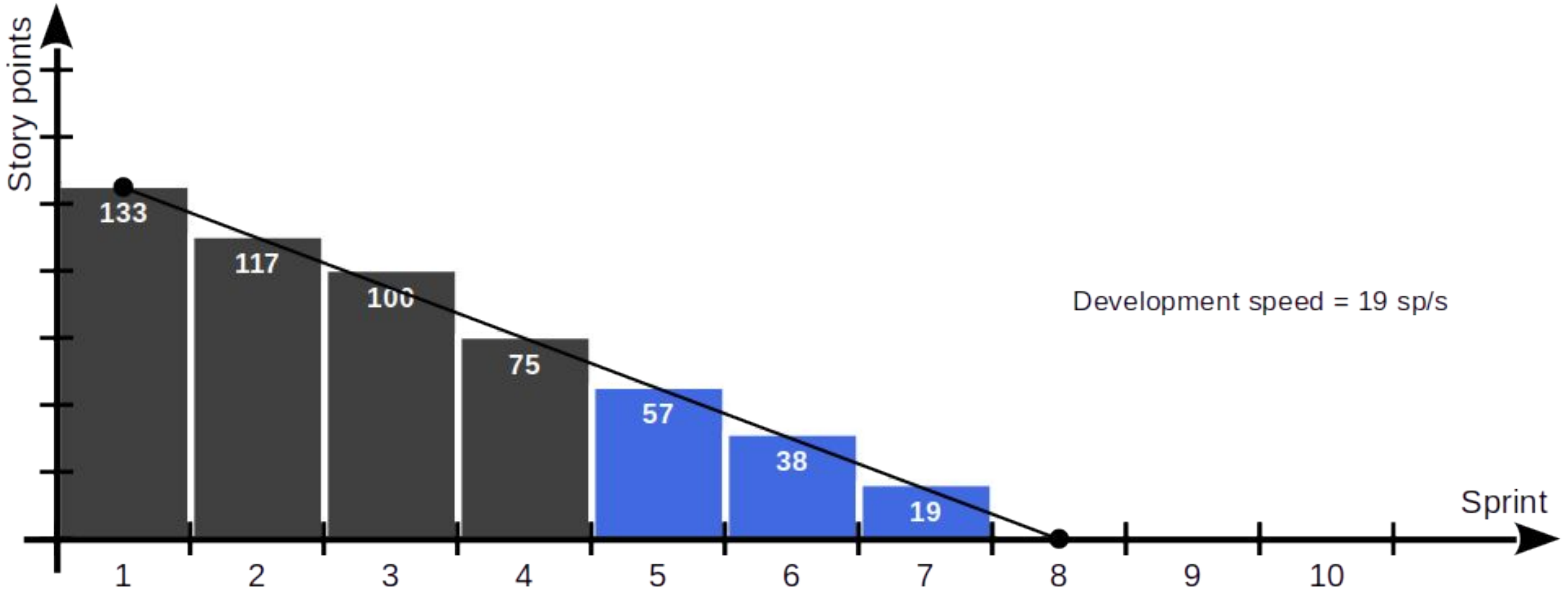
Burn-down to Project / Product Release (Burn-down Chart)



Estimated vs. Real Burn-Down



Adjusting the Release Plan to Reality



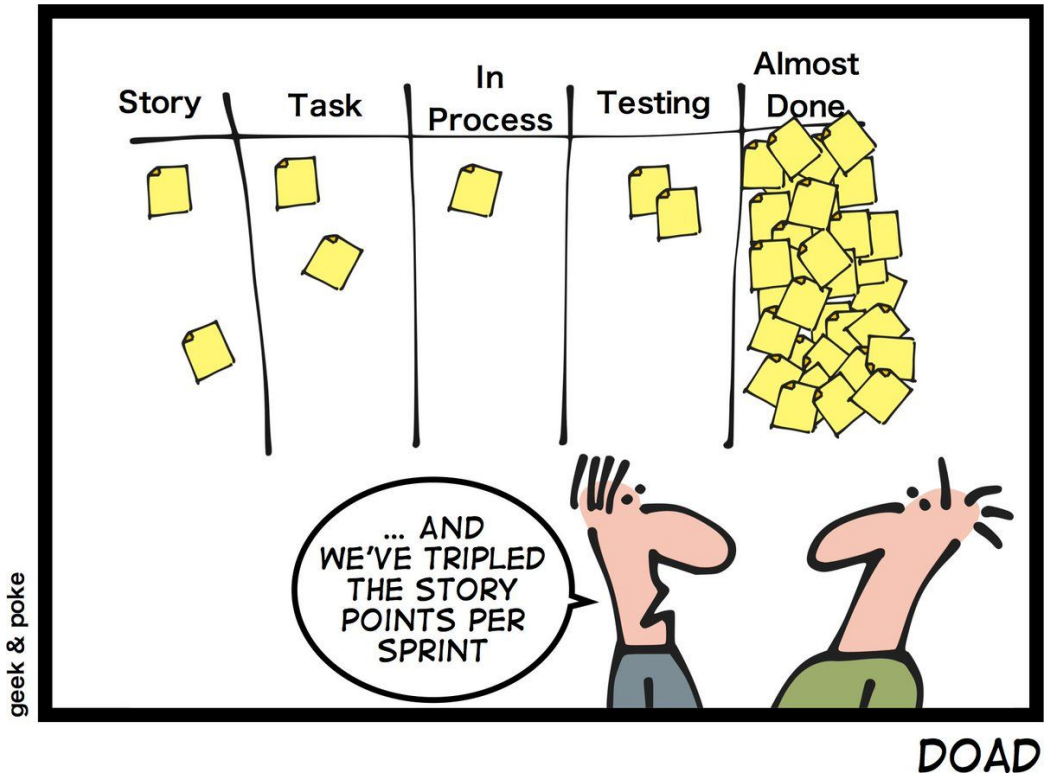
Regular Deliverable: Project Release Plans

Please create a mid-project and final release plan and keep them updated

(The initial version of the final release plan is due only after the mid-project release.)

6. Definition of Done

Almost Done …


Story
Task
In Process
Testing
Almost Done
... AND WE'VE TRIPLED THE STORY POINTS PER SPRINT
geek & poke
DOAD

Definition of Done

A **definition of done** (DoD) is

- An auditable check-list of propositions about an artifact
- Shared by all artifacts of the same type
- Typically of a technical nature

Assessing whether the artifact is “done”

Decisions Utilizing Definitions of Done

There are three main decisions with associated definitions of done

1. Feature
2. Sprint release
3. Project release

In contrast, to feature definitions of done, acceptance criteria are

- Specific to each backlog item
- Typically of an application domain nature

Example Definition of Done for Features

- Component tests have been written and pass
- Code review has been completed and code has been merged
- All feature branches have been merged and closed
- New feature code has been documented

Example Definition of Done for Sprint Release

- Project builds, deploys, and tests successfully
- Database update scripts succeed, consistency tests pass
- Sprint release notes have been written
- Change log has been updated

Example Definition of Done for Project Release

- User interaction tests pass on all major browsers
- Component test coverage is above 70%
- Design documentation has been updated
- User documentation has been updated

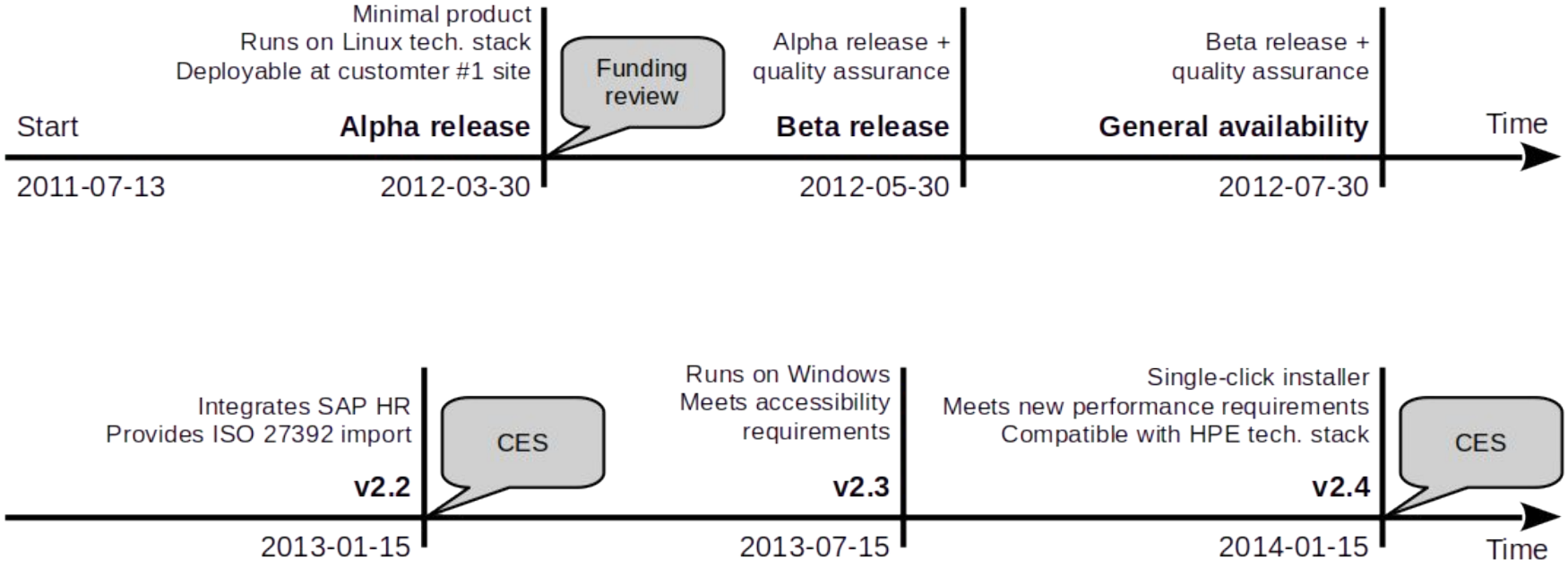
One-Time Deliverable: Definitions of Done

Please create and agree upon definitions of done for all three types

Feel free to strengthen the definitions of done over time

7. Roadmapping

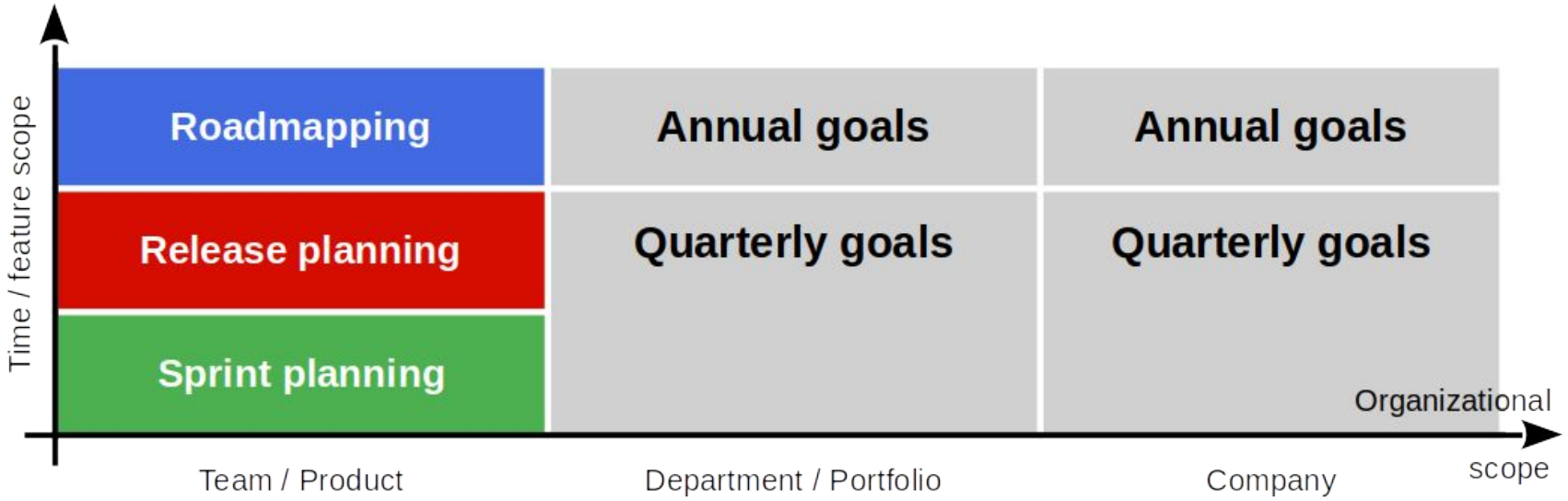
Illustration of Example Roadmap



Time Horizons of Planning Concepts

	Time-frame	Content	Certainty	Owner
Product vision	Long-term (3+ years)	High-level ideas	Low	CEO / business owner
Product roadmap	Medium (1-5 years)	Themes and epics	Medium	(Strategic) product manager
Product release	Short-term (months)	Epics and features	High	Scrum product owner

Planning vs. Organizational Scope



Summary

1. Product goal
2. Product glossary
3. Product backlog
4. Sprint planning
5. Release planning
6. Definition of done
7. Roadmapping

Thank you! Any questions?

dirk.riehle@fau.de – https://oss.cs.fau.de

dirk@riehle.org – https://dirkriehle.com – @dirkriehle

Legal Notices

License

- Licensed under the CC BY 4.0 International license

Copyright

- © Copyright 2023 Dirk Riehle, some rights reserved