Agile software development at HSLU

P. Sollberger, V 1.2

Guideline for agile software development for projects in education and research at HSLU.

Table of content

1. Introduction

agile@HSLU: Guideline for agile software development for projects in education and research at HSLU.

Students and employees at the Lucerne School of Computer Science and Information Technology are involved in various small and medium-sized projects. In some cases, these projects not only involve pure software development but also include aspects of product development.
Especially in these project assignments, lecturers and project partners not only expect the delivery of working software, but they also require an adequate and documented development process, traceable requirements, software architecture documentation, and proof that the software does what it is expected to do.

Nowadays, agility is a de facto standard for all processes. Many of the common process models used in the commercial software development industry (e.g., SAFe [1], Hermes [2], V-Model XT [3], Prince2 [4]) describe iterative and incremental development and ways to maximize customer feedback. To use these frameworks in small or medium-sized projects, they must be tailored, which requires a lot of skills and know-how about these models.
With this "agile@HSLU" guideline, the authors want to point out some key success practices, offer tips for successful project implementation and provide templates mainly for project documentation and secondary for product documentation.

This guideline does not cover all types of ICT projects. Namely infrastructure projects and projects to develop strategies are not being discussed.

2. Phases in agile projects

Agile development refers to a flexible, iterative and incremental approach that focuses on collaboration, continuous improvement and rapid adaptation to changes. The Scrum-Guide [5] describes elements and processes to successfully generate value.
Scrum however is not a project management method but a type of execution for software development within one or more phases in a project for a product development.

The agile@HSLU phase model builds on the life cycle of a product. Figure 1 shows the agile@HSLU product life cycle with phases.

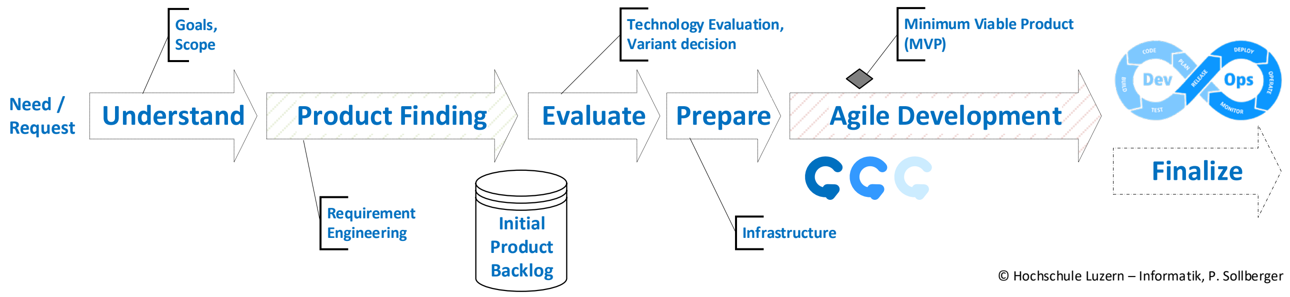


Figure 1: agile@HSLU product life cycle phases

Each development starts with a **Need/Request** expressed by a customer. This is then surveyed by several different phases.

2.1. Understand

First, project implementers must **Understand** the client's need or request: What is the problem? What are possible solutions?
The project objectives and scope must be defined and formulated together with the customer.

2.2. Product Finding

Next, start the **Product Finding** phase. Depending on the project goals, one or more methods may help to identify requirements:

- Design Thinking [6]
- Business Process Modelling [7]
- Information Modelling [8]
- CRISP-DM: CRoss Industry Standard Process for Data Mining [9]
- UX Design Process [10]
- Game Development Process [11]
- ISDP: Information security and data protection [12]
- Knowledge-creating work [13]
- …

Consider various perspectives to gain a set of requirements that is as complete as possible to be placed in the **Initial Product Backlog**. The "Product Pentagon" as shown in Figure 2 may help.

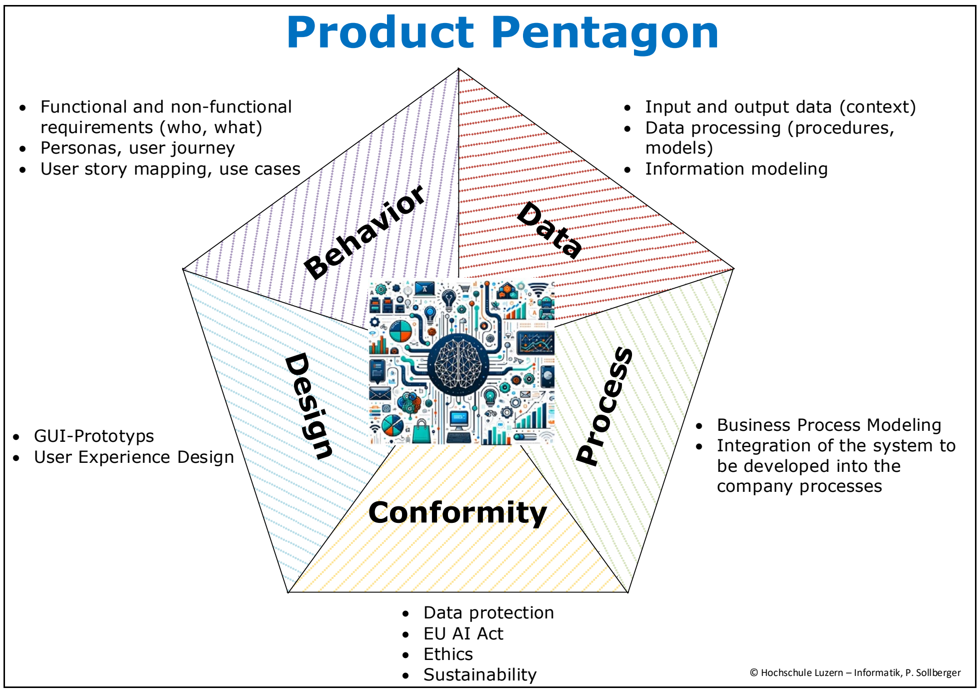

Product Pentagon
Functional and non-functional requirements (who, what)
Personas, user journey
User story mapping, use cases
Behavior
Input and output data (context)
Data processing (procedures, models)
Information modeling
Data
Business Process Modeling
Integration of the system to be developed into the company processes
Process
Data protection
EU AI Act
Ethics
Sustainability
Conformity
GUI-Prototyps
User Experience Design
Design
© Hochschule Luzern – Informatik, P. Sollberger

Figure 2: Product Pentagon - various perspectives for requirements

In an agile project, product finding resp. requirements engineering is an ongoing process and is continued in the subsequent project phases.
In Scrum, the Sprint review meeting is the time when you should assess the adequacy of the product and discuss further product finding activities.

2.3. Evaluate

Next, **Evaluate** the technology to use, weight options and decide on the best one.
When using technologies in which the team has no or little experience, the time required to learn and the difficulty of estimating efforts must be taken into account.

2.4. Prepare

"Preparation" is finished if the team can immediately start creating software once the phase has concluded. To make this possible, it is too important to refine the requirements for the backlog items selected for the first Sprint.
Further on, the **Prepare** phase is used to setup the development infrastructure and work out the test strategy.

2.5. Agile Development

During **Agile Development**, the team iteratively and incrementally creates the software in short sprints, following the Scrum method.
"Iterative" means that in each sprint, the team extends, tests, and documents the software. During the sprint, the team is also responsible to work out the detailed requirements for the following sprint (backlog refinement) and to hold the backlog refinement meeting. At the end of each sprint, the team demonstrates the working software to the customer and asks for feedback (sprint review).

“Incremental” means that the scope of the software and documentation (the number of implemented requirements) increases steadily. The team ensures that all functions always work at the end of a sprint. Test automation can be useful to achieve this.

While continuously improving the software, be careful to maintain a proper architecture. Refactoring is an ongoing, mandatory task.

2.6. Finalize

At the end of the Agile Development, either the **Finalize** phase is used to properly close the project and hand over the project results or start the transfer to combine development with operations (**DevOps**).
In many student projects, especially the project assignments in the final year (WIPRO, BAA), the project team will be graded based on the final report. It is good practice to stop software development at the end of a sprint and concentrate on finalizing the report, including all the documentation created during the agile@HSLU process.

3. agile@HSLU tailoring

Tailoring process models means selecting the artefacts and activities that are needed in the specific project and omitting aspects that are not required.

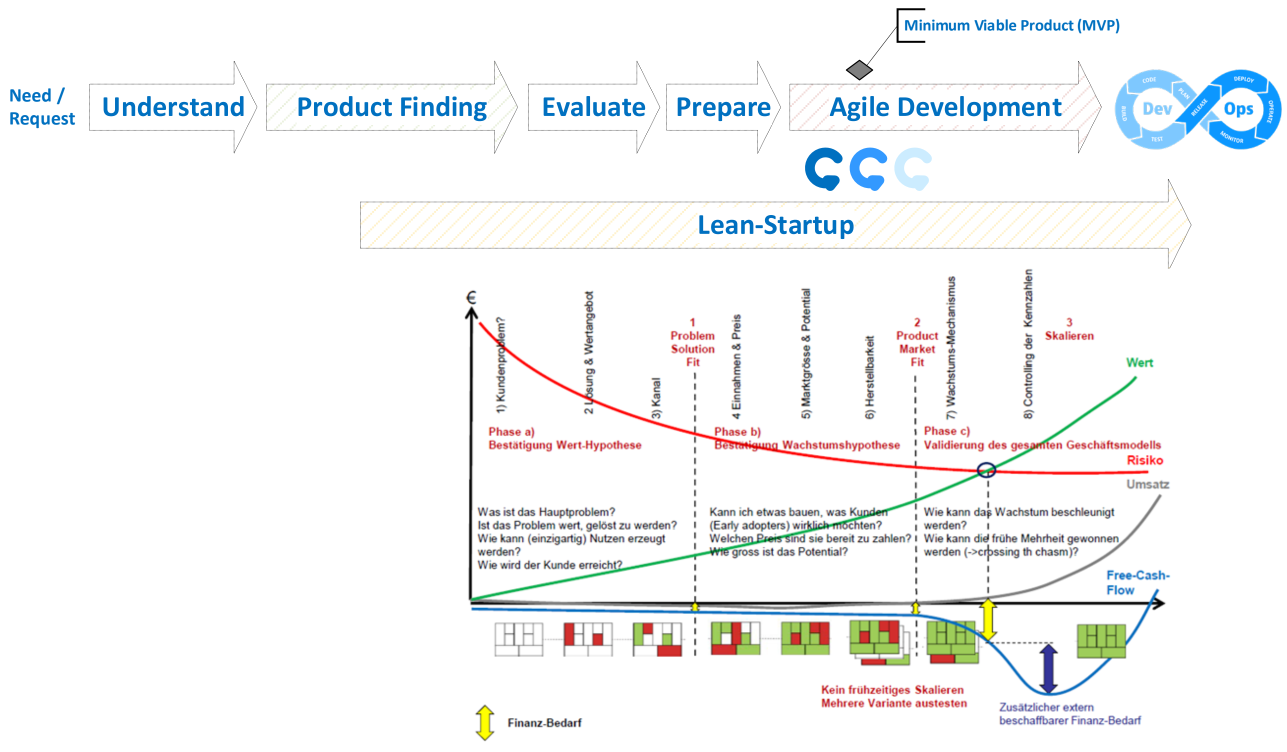
Depending on the customer need/request, once the "Product Finding" phase has been completed,

- the project work will be finalized.
- development starts to create an "Minimum Viable Product" (MVP) based on the prioritized initial product backlog and the feedback from the sprint reviews.
- continue with all following phases to fully implement the product, put it into operation and continuously develop it as required (DevOps).

If key requirements are already known at start of the project or the technology stack to be used is predetermined, the corresponding phases can either be omitted completely or they only take a short time.

3.1. Lean Startup and agile@HSLU

At the HSLU, students can set up their own business while studying. Experienced coaches assist them in the early stages (Product Finding) and help them to find out if their idea is suitable to achieve their economic objectives. That is why the HSLU offers training around the lean startup method.


Minimum Viable Product (MVP)
Need / Request
Understand
Product Finding
Evaluate
Prepare
Agile Development
Dev
Ops
Lean-Startup
€
1) Kundenproblem?
2 Lösung & Wertangebot
3) Kanal
1 Problem Solution Fit
4 Einnahmen & Preis
5) Marktgrösse & Potential
6) Herstellbarkeit
2 Product Market Fit
7) Wachstums-Mechanismus
8) Controlling der Kennzahlen
3 Skalieren
Wert
Phase a) Bestätigung Wert-Hypothese
Phase b) Bestätigung Wachstumshypothese
Phase c) Validierung des gesamten Geschäftsmodells
Risiko
Umsatz
Free-Cash-Flow
Kein frühzeitiges Skalieren Mehrere Variante austesten
Zusätzlicher extern beschaffbarer Finanz-Bedarf
Finanz-Bedarf

Was ist das Hauptproblem?
Ist das Problem wert, gelöst zu werden?
Wie kann (einzigartig) Nutzen erzeugt werden?
Wie wird der Kunde erreicht?

Kann ich etwas bauen, was Kunden (Early adopters) wirklich möchten?
Welchen Preis sind sie bereit zu zahlen?
Wie gross ist das Potential?

Wie kann das Wachstum beschleunigt werden?
Wie kann die frühe Mehrheit gewonnen werden (->crossing th chasm)?

Whitepaper Lean-Startup V0.7 / Sept. 2016, Prof. Dr. Patrick Link / Hansruedi Lingg;
https://hub.hslu.ch/smart-up/wp-content/blogs.dir/578/files/sites/12/2021/02/Whitepaper-Lean-Startup-V0.7.pdf

Figure 3: Lean Startup method integrated in the agile@HSLU product development process.

As shown in Figure 3, the **Lean-Startup** phase may run concurrently with the agile development process.

4. Tasks in agile@HSLU

4.1. Understand

Formulate the vision together with the client/customer. It will help to guide, motivate, and inspire the persons working on the project.

Next, spell out what will be better once the new product is available. According to the SMART principle [14], the formulated objectives must be specific, measurable, assignable, realistic and time related.

When defining the scope, it is negotiated with the customer which work results are to be produced by the project team and which are not their responsibility.
Also record any restrictions and constraints the project team must consider.

Set up the project organization: Identify all stakeholders and their respective roles in the project. In the development team, assign the required roles (project leader, scrum product owner, …).

Now elaborate the roadmap: Define the duration of the various phases, and if possible, determine the number and the length of the sprints in the Agile Development phase.
In education projects, the roadmap outlines the amount of work performed within the allocated time. Good practice in education projects is to include a milestone halfway into the project time and to present the successes achieved by that point.

Create a plan for the following Product Finding phase. This plan is mainly influenced by the choice of "Product Finding" method or methods. When using iterative methods: Consider using milestones as decision points for whether additional iterations are required.

Initially, identify the risks, rate them, and define, if necessary, counter measures or mitigating measures. Repeat the risk management process every iteration.

Put all results of this phase into the project management plan.

4.2. Product Finding

The Product Finding phase is used to create the "Initial Product Backlog", a collection of all known requirements so far.
In a first step, requirements are summarized in a form that is understandable to every person involved in the project (Epics). Epics are used to describe both functional and non-functional requirements.
For each requirement, formulate at least a meaningful title and a short, adequate description. Afterwards, complete each requirement with a priority and complexity: The team should implement the most valuable requirements first. The level of complexity serves as a first estimation of the implementation effort.

In a first approach, a simple table can be used for the Initial Product Backlog. Later, a tool supporting revisions may be used since all entries in the backlog improve or change over time.

4.3. Evaluate

Which technologies are suitable for the project? Use e.g., a benefit analysis: For each technology option, formulate criteria, give weight to each criterion and then rate each option.

It is good practice to use technologies that suit the team's know-how and preferences. The team can only perform well if it is familiar and comfortable with the selected option. In the subsequent development phase, there is no room to learn new technologies; it is extremely difficult to estimate the effort required to acquire new skills.

Document the evaluation in the software architecture document.

4.4. Prepare

4.4.1.Prepare - Project Planning

As soon as the initial product backlog is available, further planning activities can start. The team now has an idea of the scope of work.
Together with the team, the project leader may update the roadmap and create the release plan, a living plan that shows for the next few sprints (until the next release) when which backlog item will be implemented.
And since the product backlog changes over time (new items, changed priority, …), the release plan may later change too. Therefore, the release plan may be updated after each sprint.

The team updates the risk list (and will do so after each sprint).
The release plan might have to be updated if preventive measures (e.g., creation of a prototype) lead to additional backlog items (Spikes).

Plans and the updated risk list are put in the project management plan.

4.4.2.Prepare – Infrastructure

Set up a project repository.
If using GitLab, the content of the initial product backlog may be transferred to the «issues» system [15].

Arrange a development environment for each team member. Virtualization may help to ensure that all team members work with the same versions of the compilers, framework, libraries, test framework and so on.

Prepare the remaining documentation and agree on how to collaborate in updating it. In agile development, it is good practice to keep the full documentation up to date after each sprint. The living documentation approach supported on major platforms such as GitLab or GitHub are currently considered good practice. These platforms use the term pages.

The documentation of the development infrastructure is part of the project management plan.

4.4.3.Prepare – Requirements Engineering

Requirements engineering in the Prepare phase includes refining the backlog items that might be chosen in the first sprint. The refinement starts with the formulation of user stories with the acceptance criteria. Story-splitting is used for backlog items that are too big. Afterwards, define the detailed requirements for each user story. This can be done for example with the help of a use case description, specification by example or a story board for a graphical user interface.
Requirements engineering preparation ends with the backlog refinement meeting between the team and the client/customer. This is necessary to give the team a chance to develop the correct software from the start.

4.4.4.Prepare – Test Strategy

No testing is possible if there is a shortage of time and money. But this is not good idea. Therefore, it is important to systematically define test priorities with a suitable test strategy.

A good test strategy includes the following three steps:
1. Test focus
2. Test prioritization
3. Rough definition of tests and test types

As an artefact, the test strategy documents and justifies these three points.
1. Determine the test focus.
2. Depending on the product or project, the following questions can be answered:
   - Which legal frameworks have an influence (regulations, data protection, security, ...)?
   - In which league are we?
   - How long will the product live (lifecycle)?
   - What are the potential consequences of an error or damage (at-risk people, environment, reputation, etc.)?
   - What resources are available (time, personnel, infrastructure)?
3. Prioritization
   Appropriate prioritization ensures that the most beneficial item is tested first. The strategy determines how prioritization should be achieved.
   Often, prioritization according to risks has proven to be effective (risk-based testing). The "RPI method" is recommended as a tool.
4. Tests, types of tests
   Based on the findings from points 1 and 2, the necessary tests and test types are roughly defined and specified as a guideline in step 3. The "agile testing quadrant" [16] and the quality criteria according to ISO 9126 [17] are useful aids here.

Special attention should be paid to the test infrastructure that may be required (see also chapter «Prepare - Infrastructure»), especially for security and load tests as well as tests with sensitive data (anonymization).

These specifications are then iteratively specified in the test strategy documentation.

4.5. Agile Development

4.5.1.Agile Development – Sprints

A sprint should not last more than two weeks since one key success factor in agile development is immediate and regular feedback.

A sprint in agile development starts with a planning meeting. The team decides which backlog items will be implemented. The team may use the planning poker technique to estimate efforts.
For each selected backlog item, work must be further divided into tasks, which is a unit of work one person is able to proceed within one day. Typical tasks are: «design the …», «prepare database», «code the …», «automate test», ….
A task is only finished once the associated documentation is updated!

During the sprint, the team continuously maintains the product backlog. The refinement of backlog items for the next sprint is mandatory. In the middle of the sprint, hold the backlog refinement meeting to check if the detailed requirements for the next sprint are understood by the team, accepted by the client/customer and the acceptance criteria are written down.

In line with the progress of the project, the test scope also grows and changes with each sprint. While the focus is on «feasibility» and on functional tests that prove that something works (positive tests) in the first sprints, other aspects must be considered and, if necessary, planned and carried out in the subsequent sprints.

Here, too, the risk assessment and the ISO 9126 standard (software quality) serve as a basis for a renewed perspective and to pragmatically — or agilely, i.e., following the greatest benefit—expand the test coverage.
The following points should be given special attention:

- Design and construction of negative tests
- Design and construction of limit tests and exception tests
- Increased automation of tests
- Maintain, update, and expand the necessary test data, as it often must be anonymized (data protection) and always kept relevant and available in the correct format.
- Maintain, adapt, and extend the test infrastructure, including the necessary authorizations and licenses.

These activities should also be included in the product backlog because what is not planned will never be implemented.

At the end of each sprint, the team demonstrates the running software to the customer/client on the live system and asks for feedback. If necessary, new backlog items will be inserted.

The sprint review meeting ends with for a preview of the next sprint. Does it fit the release plan or are there any change request? And what about the risk? Are further risk reduction actions due?

During the sprint retrospective, the team identifies process improvements. It might discuss if their estimations during sprint planning were accurate, and if not, how to improve it.

In the project management plan, state the estimated effort and the actual expense for each selected backlog items. The findings from the sprint review, the retrospective and an updated risk list complete the sprint controlling documentation as part of the project management plan.

4.6. DevOps

4.6.1.Definition

Under a DevOps model, development and operations teams are no longer separated.

Sometimes, these two teams are merged into one single team where the engineers work across the entire application lifecycle. The work is from development through test, deployment to operations.

The range of skills is not limited to a single function.

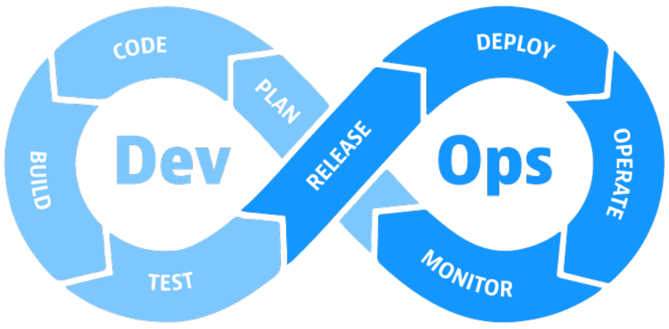


Figure 4: DevOps principle: Combining development with operations and the processes that support them enables organizations to keep pace with the speed of development.

In some DevOps models, quality assurance and security teams may also become more tightly integrated with development and operations.

When security is the focus of everyone on a DevOps team, this is sometimes referred to as DevSecOps.

These teams practice automating processes that historically have been manual and slow. They use a technology stack and tooling which help them operate and evolve applications quickly and reliably. These tools also help engineers independently carry out tasks that normally would have required help from other teams. This increases a team's velocity.

Wikipedia [17] provides the following definition:
"DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the system's development life cycle and provide continuous delivery with high software quality. DevOps is complementary with Agile software development; several DevOps aspects came from the Agile methodology."

4.6.2. Activities

DevOps teams practice daily:

- **Coding**: Code development and review, source code management tools, code merging. Emphasize continuous development and testing. Continuous testing can only be fully automated.
- **Building**: Continuous integration tools, build status. Emphasize continuous integration, continuous delivery, and continuous deployment
- **Testing**: Continuous testing tools that provide quick and timely feedback on business risks. Supports continuous development activities.
- **Packaging**: Artifact repository, application pre-deployment staging. Supports the continuous delivery activities in a corporate environment.
- **Releasing**: Change management, release approvals, release automation. Supports the continuous delivery and deployment of productive servers.
- **Configuring**: Infrastructure configuration and management, infrastructure as code tools. Supports the infrastructure as a code approach.
- **Monitoring**: Applications performance monitoring, end-user experience. Emphasize continuous monitoring and alarming, optimize time to recover metric.

Here are some of the most critical DevOps metrics:

- **Deployment Frequency**: It analyzes how frequently you are deploying the current release of software into production. Deployment automation is covered through continuous deployment and continuous delivery. Higher frequencies correlate with high-performance teams.

- Average Lead Time: It identifies how long it takes to develop, examine, deliver, and deploy a brand-new requirement through lead time tracking. Value stream approaches emphasize optimizing lead time.
- **Meantime To Recovery**: Measures the time between an interruption due to deployment or system failure and full recovery through mean time to recovery MTTR tracking. Focus is on efficient recovery and away from mean time between failure. If your organization can recover in minutes, the failure rate is seldom critical.
- **Change Failure Rate**: Indicates how often a team's changes or hotfixes lead to failures after the code has been deployed.

Microsoft has published a "DevOps checklist» [18] to assess your DevOps culture and process.

4.7. Finalize

To close a project, all remaining tasks must first be completed. This is followed by handing over the results to the client or stakeholders, along with a final approval. Lastly, a project closure report is prepared, evaluating the project and capturing key insights for future projects.

5. Artefacts in agile@HSLU

As explained in the previous chapter, all gathered information and all decisions shell be written down in one or the other artefact. Following explanations to the contents of typical software development artefacts shall serve as guideline.

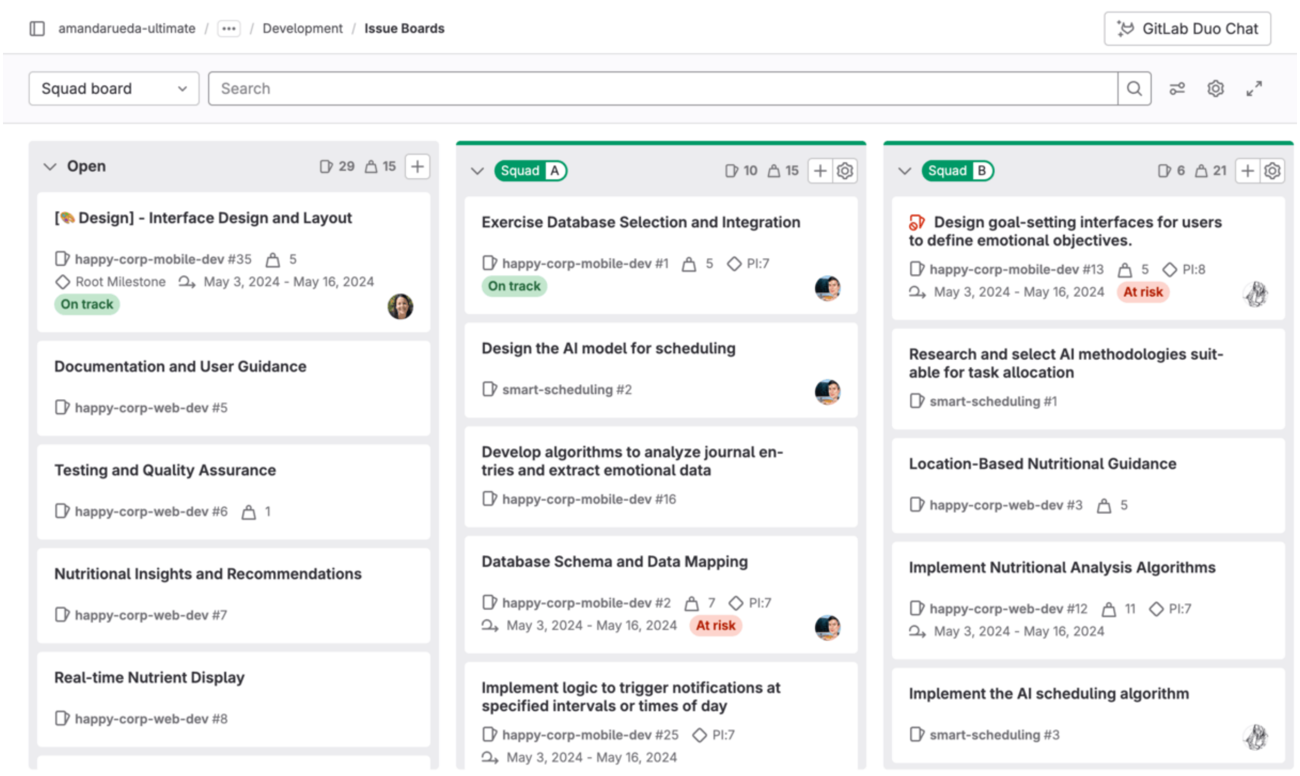
5.1. Product Backlog

In the Product Finding phase, an initial product backlog may only be represented by a simple table.

Titel *Free text, short*	**Description** *Free text*	**Benefit** *Free text*	**Provider** *Stakeholder*	***Priority*** *1 - high* *2 - medium* *3 - low*	**Complexity** *1 - low* *2 - medium* *3 - high*	**Remark** *Free text*
Requirement Group 1						
Requirement Group 2						

Figure 5: Table used for the initial Product Backlog

Later, a tool like GitLab or Jira [19] may be used to track progress over the backlog items. The next figure shows an GitLab issue board (source: GitLab Docs [20]).



Open 29 15

[Design] - Interface Design and Layout
happy-corp-mobile-dev #35 5
Root Milestone May 3, 2024 - May 16, 2024
On track

Documentation and User Guidance
happy-corp-web-dev #5

Testing and Quality Assurance
happy-corp-web-dev #6 1

Nutritional Insights and Recommendations
happy-corp-web-dev #7

Real-time Nutrient Display
happy-corp-web-dev #8

Squad A 10 15

Exercise Database Selection and Integration
happy-corp-mobile-dev #1 5 PI:7
On track

Design the AI model for scheduling
smart-scheduling #2

Develop algorithms to analyze journal entries and extract emotional data
happy-corp-mobile-dev #16

Database Schema and Data Mapping
happy-corp-mobile-dev #2 7 PI:7
May 3, 2024 - May 16, 2024 At risk

Implement logic to trigger notifications at specified intervals or times of day
happy-corp-mobile-dev #25 PI:7
May 3, 2024 - May 16, 2024

Squad B 6 21

Design goal-setting interfaces for users to define emotional objectives.
happy-corp-mobile-dev #13 5 PI:8
May 3, 2024 - May 16, 2024 At risk

Research and select AI methodologies suitable for task allocation
smart-scheduling #1

Location-Based Nutritional Guidance
happy-corp-web-dev #3 5

Implement Nutritional Analysis Algorithms
happy-corp-web-dev #12 11 PI:7
May 3, 2024 - May 16, 2024

Implement the AI scheduling algorithm
smart-scheduling #3

Figure 6: Example of a GitLab Issue Board

For members of HSLU, a training on how to use GitLab can be found on ILIAS [21].

5.2. Project Management Plan

The project management plan contains all information and decisions concerning the development process. Some content of the plan remains constant over the entire project time.
Other parts must be maintained an updated after each iteration, as for example the release plan, the risk list and the project burndown chart, other parts are unique per sprint (e.g., sprint planning and controlling, review and retrospective results).

The next subchapters describe the content of a good practice project management plan.

5.2.1.Vision and Goals

The project vision serves as the inspiration and provides focus for the project. It articulates the reason for the project in a short and understandable form, and it is important for the entire team to understand, share and work towards the same vision throughout the whole project.
A goal is a desired future state that is precisely defined in terms of content, time, and extent.
Together with the customers and clients, try to formulate at least three but not more than six goals.
Goals should be formulated as "SMART" as possible. "SMART" stand for:

• **S**pecific	The goal must apply to concretely named organizations, framework conditions, etc.; the boundaries of the target area are specified.
• **M**easurable	Qualities and, if necessary, quantities of goal achievement can be determined; indicators can be derived.
• **A**ppropriate	Is it the "right" goal? Is there a need for the planned measures?
• **R**ealistic	The prospects of achieving the goal are sufficiently high under the given framework conditions (resources, time, competencies); external, uncontrollable factors do not stand in the way of achieving the goal.
• **T**ime-bound	A time frame is given.

5.2.2.Scope

Create a list of all artefacts customers and clients expect as an output from the project team.
In a software development project, the main deliverable is the software itself. But customers and clients may wish to obtain documents describing the product, showing how the software was tested, helping them use the software, ….
Deliverables may be:

- Application software
- Software architecture documentation
- Test Strategy and Test Script Documentation
- User manual
- Deployment, installation, and maintenance manuals
- Project management plan

Customers and clients often wish to receive intermediate deliverables to start using parts of the new product as soon as possible (see chapter Roadmap).
The following table may be used to define the project deliverables:

Artefact	Release	Shortcut	Expected scope
Software	Proof of Concept	SW PoC	…
	Release 0.5	SW 0.5	
	Release 1.0	SW 1.0	
Software Architecture Documentation	Proof of Concept	SAD PoC	
	Release 0.5	SAD 0.5	
	Release 1.0	SAD 1.0	
Test Strategy and Test Script Documentation	Proof of Concept	TST PoC	
	Release 0.5	TST 0.5	
	Release 1.0	TST 1.0	
User manual	…	UM …	
…			
Project Management Plan	Release 0.1	PMP 0.1	
	Release 0.5	PMP 0.5	
	Release 1.0	PMP 1.0	

Figure 7: List of deliveries

With the list of deliveries, you clearly define what is in scope of the project. It is a good practice to clearly define also parts that are "out-of-scope".
Complete the scope description with writing down possible constraints.

5.2.3.Project Organization

Create a list with the persons involved or affected by the project (the stakeholders) and describe their role.
Setup the project team and define who takes over the role of the project leader, the scrum product owner, and the scrum master.

5.2.4.Project Support

Describe (e.g., in the form of an installation instruction) how to set up the development environment. What tools, servers, databases, frameworks, and libraries are used for development and testing? Outline the repository organization.

5.2.5.Definition of Ready, Definition of Done

Next to the deliverables, define quality aspects the team meets when implementing a backlog item.

With the Definition of Ready (DoR), the team formulates criteria for the backlog refinement activities: What details are required before a backlog item is ready to be implemented in the next sprint.

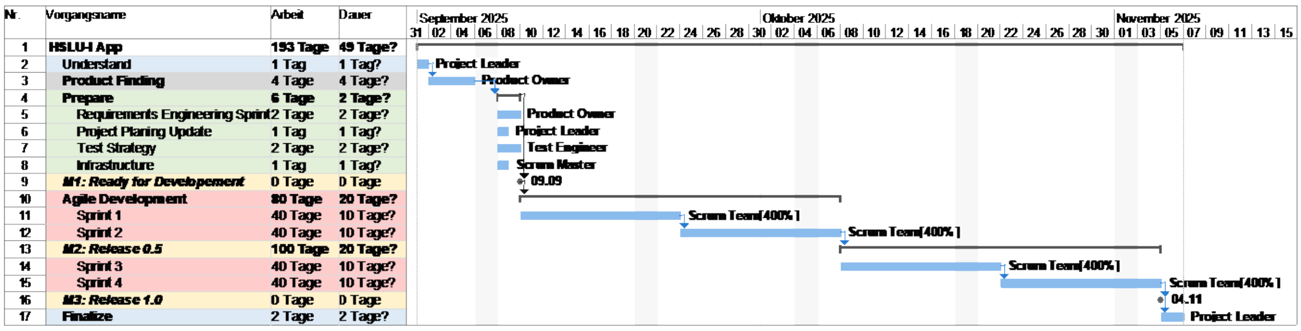
The description of the Definition of Done (DoD) states when a product backlog item is completed. This DoD is a contract between the developers and customers and ensures everyone in the team knows exactly what is expected. It ensures transparency and quality fit for the purposes of product and organization.
In software, a Definition of Done may be: "Done means coded to standards, reviewed, implemented with unit Test-Driven Development (TDD), tested with 100 percent test automation (or according to the test strategy), integrated, and documented."

5.2.6.Roadmap

The roadmap is the long-term planning of the project and is used to check the economic efficiency.
Key elements of the roadmap are phases, milestones and the Sprints in the "Agile Development" phase.

A milestone is a planned point in the project process at which predefined, measurable [interim] deliveries are available that allow to determine the project process and project costs.
To each milestone, define which deliveries shall be available.
The milestone is reached if the required deliveries are available, and their review was successful.

A roadmap with three milestones may look like this.



Nr.	Vorgangsname	Arbeit	Dauer
1	HSLU-I App	193 Tage	49 Tage?
2	Understand	1 Tag	1 Tag?
3	Product Finding	4 Tage	4 Tage?
4	Prepare	6 Tage	2 Tage?
5	Requirements Engineering Sprint	2 Tage	2 Tage?
6	Project Planing Update	1 Tag	1 Tag?
7	Test Strategy	2 Tage	2 Tage?
8	Infrastructure	1 Tag	1 Tag?
9	*M1: Ready for Developement*	0 Tage	0 Tage
10	Agile Development	80 Tage	20 Tage?
11	Sprint 1	40 Tage	10 Tage?
12	Sprint 2	40 Tage	10 Tage?
13	*M2: Release 0.5*	100 Tage	20 Tage?
14	Sprint 3	40 Tage	10 Tage?
15	Sprint 4	40 Tage	10 Tage?
16	*M3: Release 1.0*	0 Tage	0 Tage
17	Finalize	2 Tage	2 Tage?

Milestone		Deliveries
M1	Start agile development	PMP 1.0
M2	Release 0.5	SW 0.5
		SAD 0.5
	Describe here the scope …	TST 0.5
		PMP 1.2
M2	Release 1.0	SW 1.0
		UM 1.0
	Describe here the scope …	SAD 1.0
		TST 1.0
		PMP 1.4

Figure 8: Roadmap with phases, milestones and sprints

5.2.7.Release Plan

The release plan in agile development is the short-term plan and states what is available after the next few sprints. It is based on the prioritized entries of the actual product backlog and shows which backlog items will be implemented in which of the following sprints.
The release plan can be a simple table or several issue boards in GitLab.

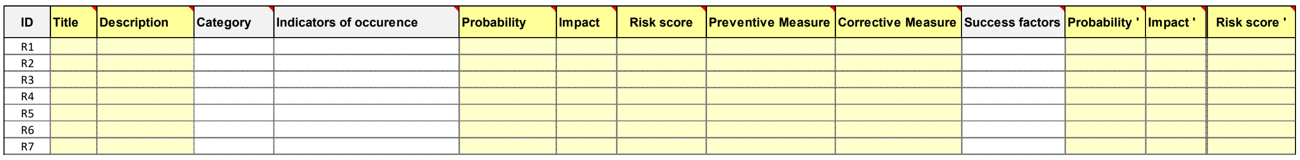
Titel	Description	Priority	Complexity	Estimated Effort	Sprint objectives
Free text	*Free text*	*1 - high*	*1 - low*	*amount of work*	*Free text*
		2 - medium	*2 - medium*	*in person days*	
		3 - low	*3 - high*		
Pers. Stundenplan	Persönlicher Stundenplan für aktuellen Tag mit Rauminfo (inkl. Umbuchungen)	1	2	5	Sprint 1: Tagesaktualitäten
Tages-Menuplan	…	1	2	3	
Anlässe	Aktuelle Anlässe	1	1	2	
Wochenplan	Persönlicher Stundenplan für aktuellen Woche, graphisch	2	2	5	Sprint 2: Wochenaktualitäten
Mitteilungen zum Studium	Übersicht inkl. Eintrag in Tages- und Wochenplan von Infoevents zum Studium.	2	2	3	
Wochen-Menuplan	…	2	2	3	
Specials	Wochenhits, z.B. Wildsaison etc.	2	3	2	Sprint 3: Aktuelle Mitteilungen
Infos	Aktuelle Infos	2	1	7	
Link zu Unterrichtsunterlagen	Link zu ILIAS, direkt aus Stundenplan aufrufbar	3	1	3	

Figure 9: Example of a release plan as a simple table

The release plan is a living plan. This means, after each sprint, the release plan is updated, and the content of next sprint will be added. Keep a snapshoot of the release plan for each iteration.

5.2.8.Risk Management

The purpose of risk management is to identify potential problems before they occur and then to take actions and implement measures throughout the project life cycle to ensure that these problems do not occur and endanger the project objectives.
To identify risks, use checklists, make a brainstorming with stakeholders, and use experiences from former projects.
Describe each risk and assess the probability of occurrence and the impact. For the highest risks (see risk matrix), identify indicators of occurrence and elaborate preventive action to mitigate the probability and/or corrective action to reduce the resulting damage. Next, try to estimate the effect of the actions.
To help gathering all this information, the following table may be used.



ID	Title	Description	Category	Indicators of occurence	Probability	Impact	Risk score	Preventive Measure	Corrective Measure	Success factors	Probability '	Impact '	Risk score '
R1													
R2													
R3													
R4													
R5													
R6													
R7													

Figure 10: Table used for risk management.

To visualize the actual risk situation and the effects of your risk management actions, use risk matrices.

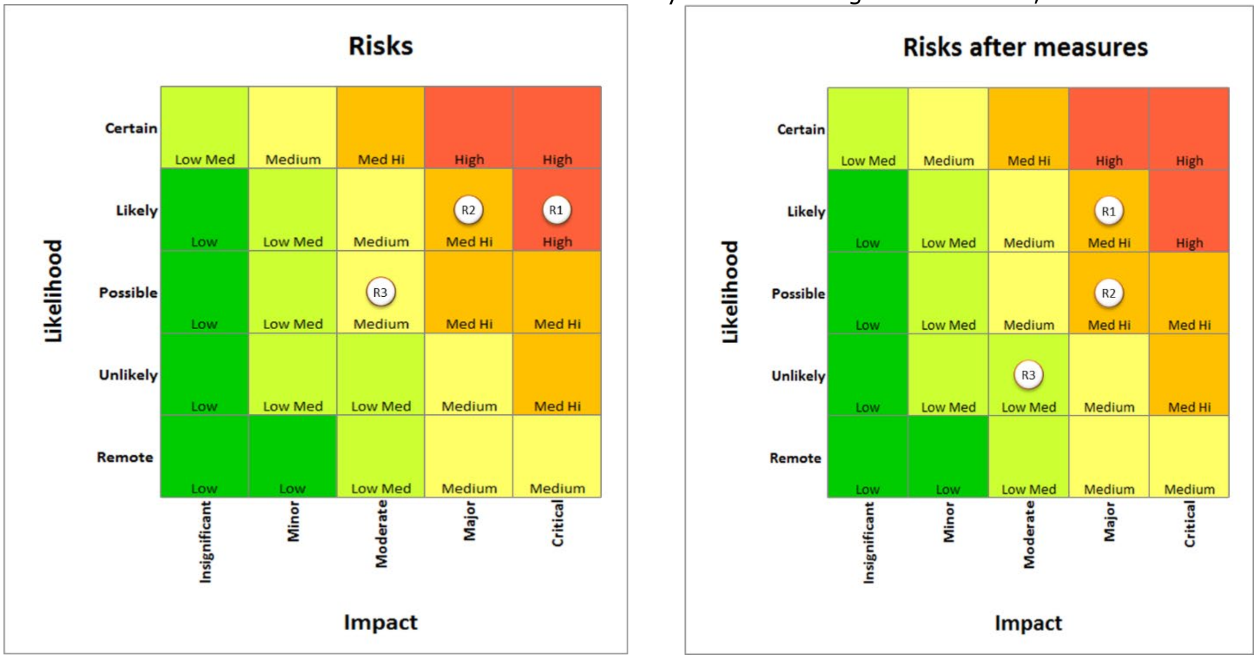


Figure 11: Risk matrices

It is important to check whether the risks recorded and described in risk management match those mentioned in the test strategy. The aim is to ensure that "risk-based testing" matches the project risks.

Keep a snapshoot of the risk list for each iteration.

5.2.9.Controlling

Project control includes all activities to detect project-related deviations between the planned and actual status.

Reasons for deviations may are:

- Changed or new requirements increases project effort.
- Effort estimation errors in Sprint planning
- Misunderstood requirements
- Occurrence of risks
- Inefficient teamwork

To detect the above deviations, there exists for each reason a tool.

Changed or new requirements

The **project burndown chart** shows the remaining amount of work as contained in the product backlog over time. The sprints are used as time ticks on the x-axis.
The trend line on the project burndown chart will generally trend downward. However, if new items are added to the backlog, then the total points remaining may go up.

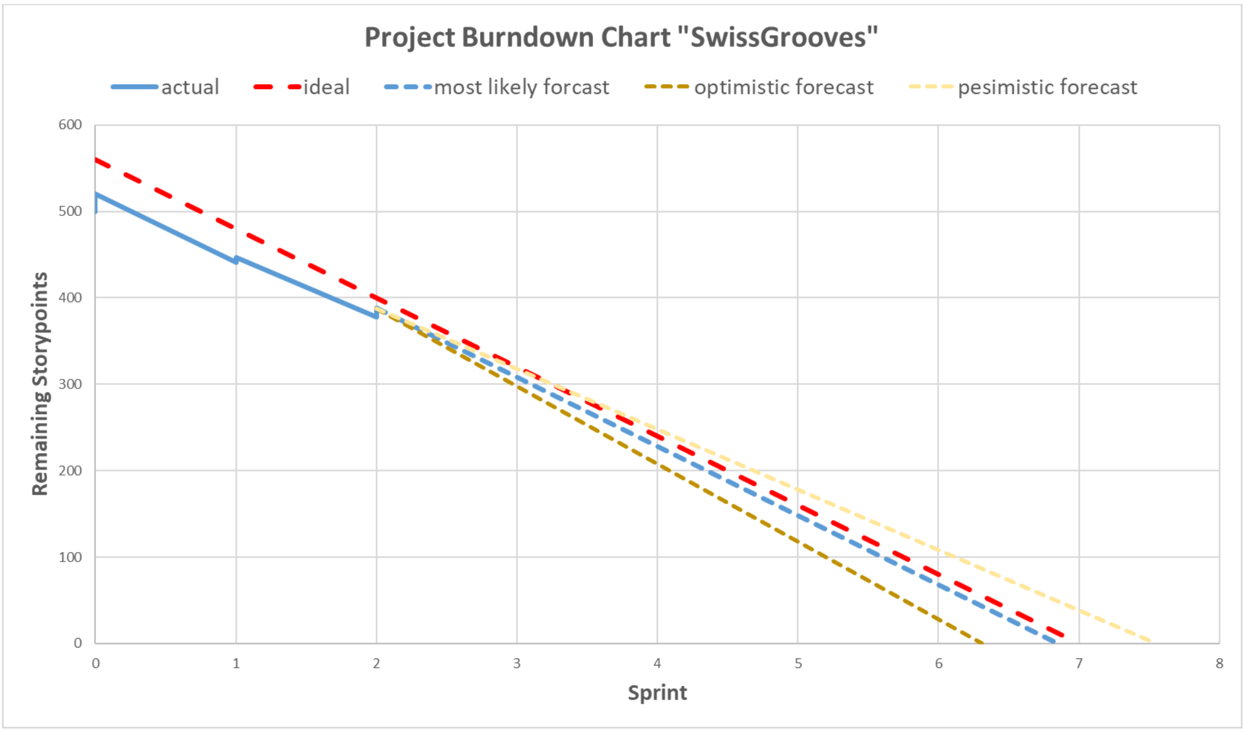


Figure 12: Project burndown chart

In literature, the project burndown chart is often referred to as release burndown chart. The term "release" there is used to paraphrase the deliveries for a milestone.

Effort estimation errors in Sprint planning

To improve the team’s capability in effort estimation in Sprint planning, use for each sprint a simple table to **compare estimations with actual effort**. Of course, it is necessary that each team member notes down the work done per task or user story to get real actual values.

User story title	Efforts in person days			Remarks
	Estimated	Actual	Deviation	
…	5	7	-2	
Total	**5**	7	-2	

Figure 13: Sprint efforts retrospective

Reflect these results in the team (→ Retrospective)

Misunderstood requirements
At the end of a Sprint, the team presents the outcome within the framework of the **Sprint review** to the stakeholders. The product backlog may also be adjusted to meet new opportunities.
Report for each Sprint the findings of the Sprint review.

Occurrence of risks
Risks are monitored periodically as part of project controlling. If necessary, a risk analysis is carried out again, e.g., in the case of new backlog items or major changes in the project.
Report for each Sprint traceable all **changes in the project risk list**.

Inefficient teamwork
The purpose of the **Sprint retrospective** is to review how the past sprint went in terms of the people, relationships, processes, and tools involved.
Report for each Sprint the findings of the retrospective.

5.3. Software Architecture Documentation

The software architecture document describes all aspects of the product.
A good structure to do this is described within the arc42 initiative by Gernot Starke, Peter Hruschka and Ralf D. Müller [22].
An overview of the chapter structure of arc42 software architecture document is given in the next figure.

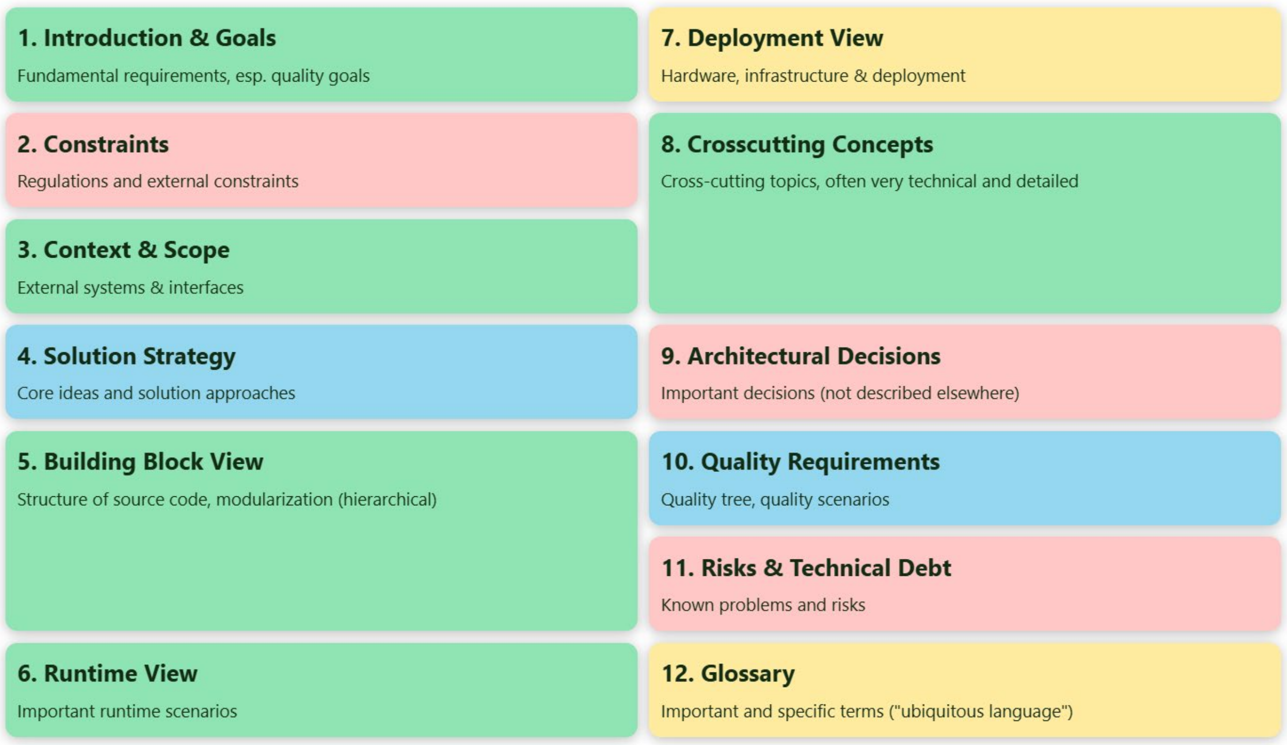

1. Introduction & Goals
Fundamental requirements, esp. quality goals
2. Constraints
Regulations and external constraints
3. Context & Scope
External systems & interfaces
4. Solution Strategy
Core ideas and solution approaches
5. Building Block View
Structure of source code, modularization (hierarchical)
6. Runtime View
Important runtime scenarios
7. Deployment View
Hardware, infrastructure & deployment
8. Crosscutting Concepts
Cross-cutting topics, often very technical and detailed
9. Architectural Decisions
Important decisions (not described elsewhere)
10. Quality Requirements
Quality tree, quality scenarios
11. Risks & Technical Debt
Known problems and risks
12. Glossary
Important and specific terms ("ubiquitous language")

Figure 14: Structure of the software architecture documentation according arc42

The arc42 website https://arc42.org/ explains in detail the structure used to construct, communicate and document the software architecture.

Next to detailed explanation of the expected content of the various chapters, the website contains tips and real-world examples.

6. Directories

6.1. Bibliography

[1] „SAFe 6.0 Framework", Scaled Agile Framework. Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://scaledagileframework.com/

[2] „HERMES-Projektmanagement-Methodenelemente". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.hermes.admin.ch/de/pjm-2022/verstehen/hermes-projektmanagement-methodenelemente.html

[3] „V-Modell XT Bund Version 2.4", Der Beauftragte der Bundesregierung für Informationstechnik. Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.cio.bund.de/SharedDocs/downloads/Webs/CIO/DE/digitaler-wandel/architekturen-standard/v_modell_xt_bund_pdf.pdf?__blob=publicationFile&v=6

[4] „PRINCE2 Agile® wiki". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://prince2agile.wiki/

[5] „Home | Scrum Guides". Zugegriffen: 23. Oktober 2024. [Online]. Verfügbar unter: https://scrumguides.org/

[6] J. Hehn, *Design Thinking for Software Engineering: Creating Human-Oriented Software-intensive Products and Services.* in Progress in IS Ser. Cham: Springer International Publishing AG, 2022.

[7] „BPMN Specification - Business Process Model and Notation". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.bpmn.org/

[8] „What Is Data Modeling? | IBM". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.ibm.com/topics/data-modeling

[9] „CRISP DM: Das Modell einfach erklärt (mit Infografik)". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.kobold.ai/crisp-dm/

[10] T. Green und J. Labrecque, *A Guide to UX Design and Development: Developer's Journey Through the UX Process*, 1st ed. 2023. in Design Thinking. Berkeley, CA: Apress L. P, 2023. doi: 10.1007/978-1-4842-9576-2.

[11] „What Are The Main Stages Of Game Development? | GameMaker". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://gamemaker.io/en/blog/stages-of-game-development

[12] „Information security and data protection". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.hermes.admin.ch/hermes5/en/project-management/understanding/modules/information-security-and-data-protection.html

[13] H. Balzert, M. Schröder, und C. Schäfer, „Wissenschaftliches Arbeiten - Ethik, Inhalt & Form wiss. Arbeiten, Handwerkszeug, Quellen, Projektmanagement, Präsentation, 3. Auflage", 2022, Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://dl.gi.de/handle/20.500.12116/38673

[14] Andrea, „SMARTe Ziele: Wie funktioniert die SMART-Formel?", Projekte leicht gemacht. Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://projekte-leicht-gemacht.de/blog/methoden/projektziele/die-smart-formel/

[15] „How to use GitLab for Agile software development". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://about.gitlab.com/blog/2018/03/05/gitlab-for-agile-software-development/

[16] „Testing Quadrants". Zugegriffen: 29. August 2025. [Online]. Verfügbar unter: https://www.pmi.org/disciplined-agile/agile/testingquadrants

[17] „ISO/IEC 9126", *Wikipedia*. 19. Juni 2025. Zugegriffen: 28. August 2025. [Online]. Verfügbar unter: https://de.wikipedia.org/w/index.php?title=ISO/IEC_9126&oldid=257157467

[18] „Wikipedia DevOps", *Wikipedia*. 16. September 2024. Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=DevOps&oldid=1245953239

[19] claytonsiemens77, „Recommendations for fostering DevOps culture - Microsoft Azure Well-Architected Framework". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://learn.microsoft.com/en-us/azure/well-architected/operational-excellence/devops-culture

[20] „Jira | Issue & Project Tracking Software | Atlassian". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://www.atlassian.com/software/jira

[21] „Issue boards | GitLab". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://docs.gitlab.com/ee/user/project/issue_board.html

[22] „Inhalt: GIT: HSLU ILIAS". Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://elearning.hslu.ch/ilias/ilias.php?baseClass=ilrepositorygui&ref_id=5207233

[23] D. G. Starke, „arc42 Template Overview", arc42. Zugegriffen: 24. Oktober 2024. [Online]. Verfügbar unter: https://arc42.org/overview

6.2. List of figures