Agile Softwareentwicklung an der HSLU

P. Sollberger, V 1.2

Leitfaden für agile Softwareentwicklung für Projekte in Lehre und Forschung an der HSLU.

Inhaltsverzeichnis

1. Einleitung

agile@HSLU: Leitfaden für agile Softwareentwicklung für Projekte in Lehre und Forschung an der HSLU.

Studierende und Mitarbeitende der Hochschule Luzern – Informatik sind an verschiedenen kleinen und mittelgrossen Projekten beteiligt. In einigen Fällen geht es dabei nicht nur um die reine Softwareentwicklung, sondern auch um Aspekte der Produktentwicklung.
Gerade bei Projekteinsätzen erwarten Dozenten und Projektpartner nicht nur die Lieferung einer funktionierenden Software, sondern sie verlangen auch einen adäquaten und dokumentierten Entwicklungsprozess, nachvollziehbare Anforderungen, eine Dokumentation der Softwarearchitektur und einen Nachweis, dass die Software das tut, was von ihr erwartet wird.

Agilität ist heutzutage ein de-facto-Standard für alle Prozesse. Viele der gängigen Vorgehensmodelle, die in der kommerziellen Softwareentwicklungsbranche verwendet werden (z. B. SAFe [1]Hermes [2], V-Modell XT [3], Prince2 [4]) beschreiben die iterative und inkrementelle Entwicklung und Möglichkeiten zur Maximierung des Kundenfeedbacks. Um diese Frameworks in kleinen oder mittelgrossen Projekten einsetzen zu können, müssen sie massgeschneidert werden, was viel Geschick und Know-how über diese Modelle erfordert.
Mit diesem "agile@HSLU"-Leitfaden wollen die Autoren einige wichtige Erfolgspraktiken aufzeigen, Tipps für eine erfolgreiche Projektumsetzung geben und Vorlagen vor allem für die Projektdokumentation und sekundär für die Produktdokumentation zur Verfügung stellen.

Diese Leitlinie gilt nicht für alle Arten von ICT-Projekten. Infrastrukturprojekte und Projekte zur Strategieentwicklung werden hier nicht diskutiert.

2. Phasen in agilen Projekten

Agile Entwicklung bezieht sich auf einen flexiblen, iterativen und inkrementellen Ansatz, der sich auf Zusammenarbeit, kontinuierliche Verbesserung und schnelle Anpassung an Veränderungen konzentriert. Der Scrum-Guide [5] Beschreibt Elemente und Prozesse, um erfolgreich Wert zu generieren.
Scrum ist jedoch keine Projektmanagement-Methode, sondern eine Art der Durchführung für die Softwareentwicklung innerhalb einer oder mehrerer Phasen in einem Projekt.

Das agile@HSLU-Phasen-Modell baut auf dem Lebenszyklus eines Produkts auf. Abbildung 1 zeigt den agile@HSLU Produktlebenszyklus mit Phasen.

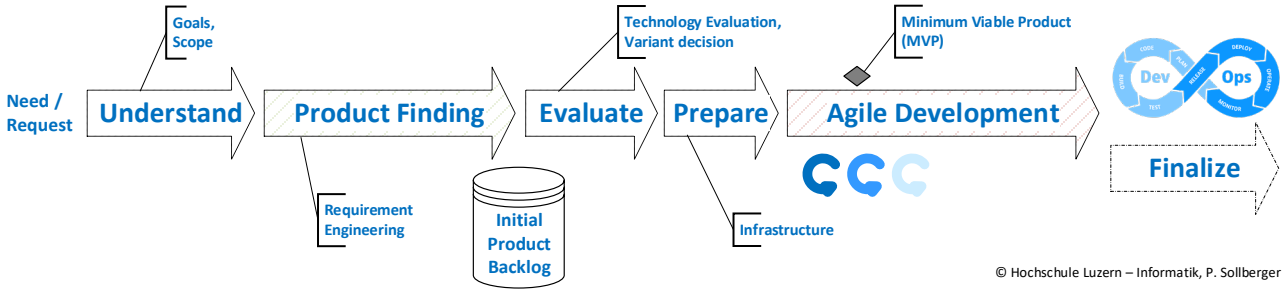


Abbildung 1: agile@HSLU Phasen des Produktlebenszyklus

Jede Entwicklung beginnt mit einem **Bedürfnis/einer Anfrage,** die von einem Kunden geäussert wird. Diese wird dann in mehreren verschiedenen Phasen untersucht.

2.1. Verstehen (understand)

Zuerst müssen Projektumsetzer den Bedarf oder die Anfrage des Kunden verstehen: Was ist das Problem? Was sind mögliche Lösungen?
Die Projektziele und der Projektumfang müssen gemeinsam mit dem Kunden definiert und formuliert werden.

2.2. Produktfindung (product finding)

Starten Sie als Nächstes die Phase der Produktfindung. Abhängig von den Projektzielen können eine oder mehrere Methoden helfen, Anforderungen zu identifizieren:

- Design Thinking [6]
- Geschäftsprozess-Modellierung [7]
- Informationsmodellierung [8]
- CRISP-DM: CRoss Industry Standard Process for Data Mining [9]
- UX-Design-Prozess [10]
- Game Development Process [11]
- ISDP: Information security and data protection [12]
- Wissensschaffendes Arbeiten [13]
- …

Berücksichtigen Sie verschiedene Perspektiven, um einen möglichst vollständigen Satz von Anforderungen zu erhalten, die im **initialen Product Backlog** abgelegt werden. Das "Produkt Pentagon", wie es in Abbildung 2 dargestellt ist, kann helfen.

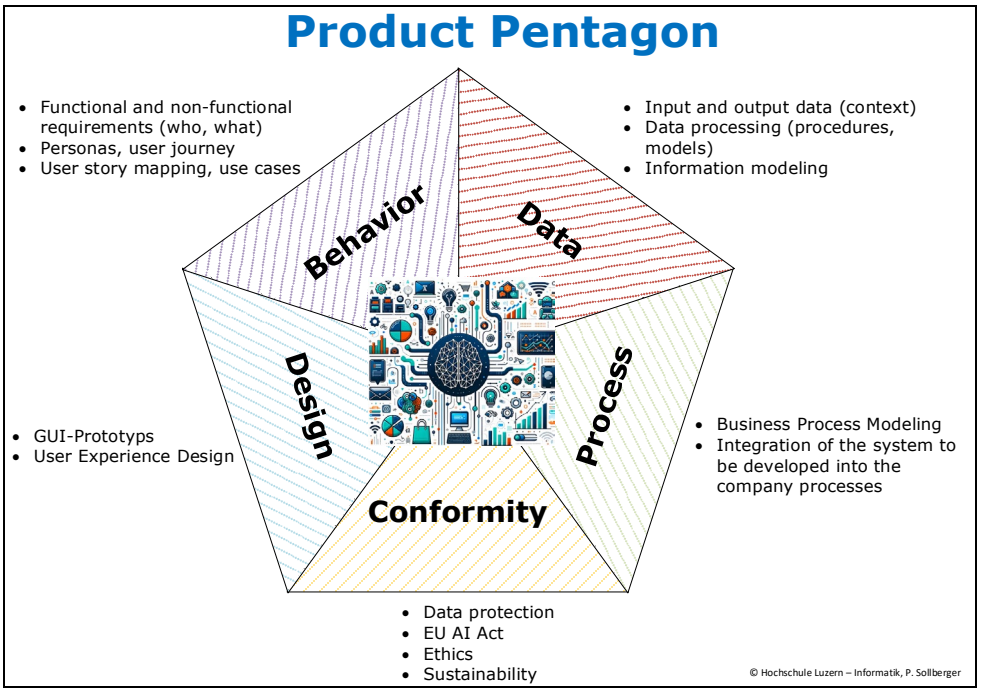


Abbildung 2: Produkt Pentagon - verschiedene Perspektiven für Anforderungen

In einem agilen Projekt ist die Produktfindung bzw. das Requirements Engineering ein fortlaufender Prozess, der in den nachfolgenden Projektphasen fortgeführt wird.
In Scrum ist das Review-Meeting der Zeitpunkt, an dem Sie die Zweckmässigkeit des Produkts beurteilen und weitere Aktivitäten zur Produktfindung besprechen sollten.

2.3. Evaluieren (evaluate)

Bewerten Sie als Nächstes die zu verwendenden Technologien, gewichten Sie die Optionen und entscheiden Sie sich für die beste.
Beim Einsatz von Technologien, in denen das Team keine oder nur wenig Erfahrung hat, müssen der Zeitaufwand für das Lernen und die Schwierigkeit der Schätzung des Aufwands berücksichtigt werden.

2.4. Vorbereitung (prepare)

Die "Vorbereitung" ist abgeschlossen, wenn das Team nach Abschluss der Phase sofort mit der Erstellung der Software beginnen kann. Um dies zu ermöglichen, ist es wichtig, die Anforderungen der Backlog-Elemente, die für den ersten Sprint ausgewählt wurden, zu verfeinern.
Im Weiteren wird die Vorbereitungsphase genutzt, um die Entwicklungsinfrastruktur aufzubauen und die Teststrategie auszuarbeiten.

2.5. Agile Entwicklung (agil development)

Bei der agilen Entwicklung erstellt das Team die Software iterativ und inkrementell in kurzen Sprints, nach der Scrum-Methode.
"Iterativ" bedeutet, dass das Team in jedem Sprint die Software erweitert, testet und dokumentiert.
Während des Sprints ist das Team auch dafür verantwortlich, die detaillierten Anforderungen für den folgenden Sprint zu erarbeiten (Backlog Refinement) und das Backlog Refinement Meeting abzuhalten. Am

Ende eines jeden Sprints demonstriert das Team dem Kunden die funktionierende Software und bittet um Feedback (Sprint-Review).
"Inkrementell" bedeutet, dass der Umfang der Software und Dokumentation (die Anzahl der umgesetzten Anforderungen) stetig zunimmt. Das Team sorgt dafür, dass alle Funktionen am Ende eines Sprints immer funktionieren. Um dies zu erreichen, kann eine Testautomatisierung sinnvoll sein.

Achten Sie bei der kontinuierlichen Verbesserung der Software darauf, eine saubere Architektur beizubehalten. Das Refactoring ist eine fortlaufende, obligatorische Aufgabe.

2.6. Abschliessen (finalize)

Am Ende der agilen Entwicklung wird entweder die Finalize-Phase genutzt, um das Projekt ordnungsgemäss abzuschliessen und die Projektergebnisse zu übergeben oder der Transfer gestartet, um die Entwicklung mit dem Betrieb zu verbinden (**DevOps**).
In vielen studentischen Projekten, insbesondere bei den Projektarbeiten im letzten Studienjahr (WIPRO, BAA), wird das Projektteam anhand des Abschlussberichts benotet. Es empfiehlt sich, die Softwareentwicklung am Ende eines Sprints zu stoppen und sich auf die Fertigstellung des Berichts zu konzentrieren, einschliesslich der gesamten Dokumentation, die während des agile@HSLU Prozesses erstellt wurde.

3. agile@HSLU Schneiderei

Massgeschneidertes Vorgehensmodelle bedeutet, die Artefakte und Aktivitäten auszuwählen, die im konkreten Projekt benötigt werden, und Aspekte wegzulassen, die nicht benötigt werden.

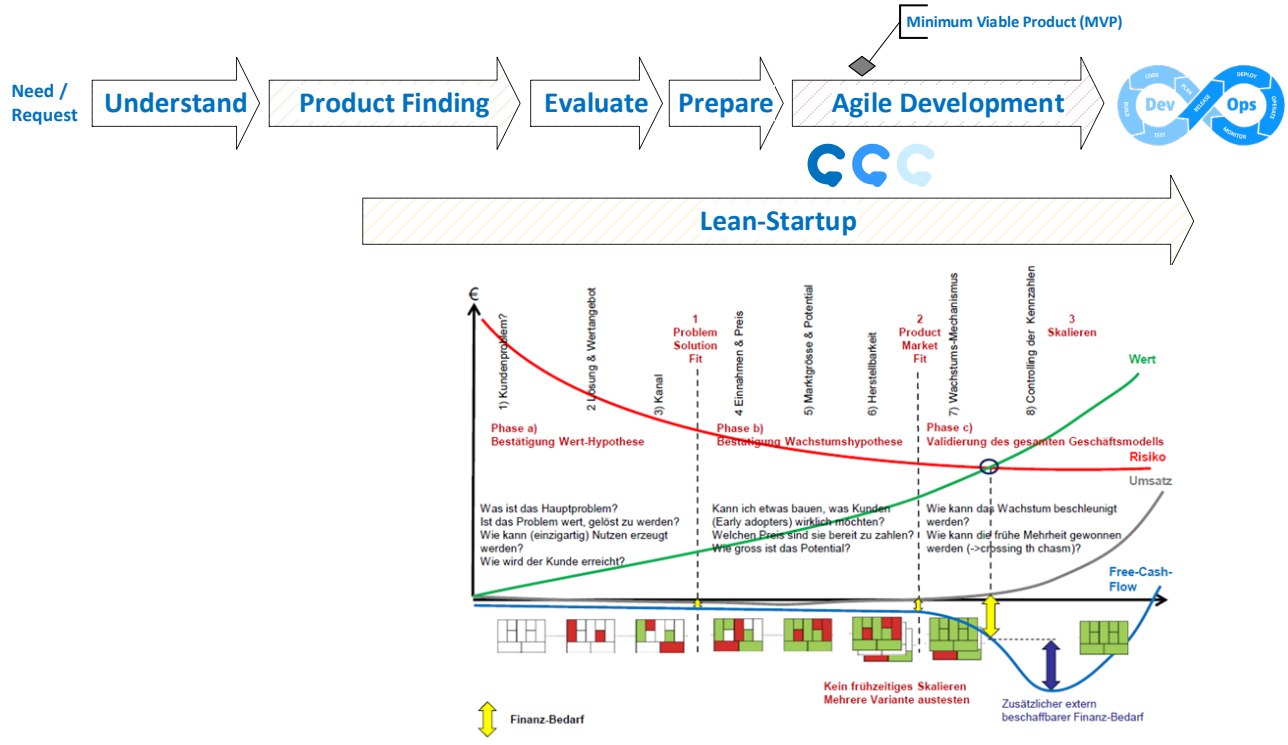
Abhängig vom Kundenbedarf/-wunsch wird nach Abschluss der Phase der "Produktfindung"

- die Projektarbeit wird abgeschlossen.
- die Entwicklung beginnt mit der Erstellung eines "Minimum Viable Product" (MVP) auf Basis des priorisierten, initialen Product Backlogs und des Feedbacks aus den Sprint-Reviews.
- mit allen folgenden Phasen fortgefahren, um das Produkt vollständig zu implementieren, in Betrieb zu nehmen und bei Bedarf kontinuierlich weiterzuentwickeln (DevOps).

Sind zu Beginn des Projekts bereits zentrale Anforderungen bekannt oder der einzusetzende Technologie-Stack vorgegeben, können die entsprechenden Phasen entweder ganz wegfallen oder sie nehmen nur kurze Zeit in Anspruch.

3.1. Lean Startup und agile@HSLU

An der HSLU können sich Studierende während des Studiums selbstständig machen. Erfahrene Coaches begleiten sie in der Frühphase (Product Finding) und helfen ihnen herauszufinden, ob ihre Idee geeignet ist, ihre wirtschaftlichen Ziele zu erreichen. Deshalb bietet die HSLU Schulungen rund um die Lean-Startup-Methode an.



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

Abbildung 3: Lean-Startup-Methode, die in den agile@HSLU Produktentwicklungsprozess integriert ist.

Wie in Abbildung 3 gezeigt, kann die Lean-Startup Phase parallel zum agilen Entwicklungsprozess ablaufen.

4. Aufgaben in agile@HSLU

4.1. Verstehen (understand)

Formulieren Sie die Vision gemeinsam mit dem Auftraggeber. Sie wird dazu beitragen, die Personen, die an dem Projekt arbeiten, anzuleiten, zu motivieren und zu inspirieren.

Legen Sie als Nächstes dar, was besser sein wird, sobald das neue Produkt verfügbar ist. Nach dem SMART-Prinzip [14]müssen die formulierten Ziele spezifisch, messbar, attraktiv, realistisch und terminiert sein.

Bei der Definition des Umfangs wird mit dem Kunden verhandelt, welche Arbeitsergebnisse vom Projektteam produziert werden sollen und welche nicht in seinem Verantwortungsbereich liegen. Notieren Sie auch alle Rahmenbedingungen und Einschränkungen, die das Projektteam berücksichtigen muss.

Einrichten der Projektorganisation: Identifizieren Sie alle Beteiligten und ihre jeweiligen Rollen im Projekt. Vergeben Sie im Entwicklungsteam die erforderlichen Rollen (Projektleiter, Scrum Product Owner, ...).

Erarbeiten Sie nun die Roadmap: Definieren Sie die Dauer der verschiedenen Phasen und bestimmen Sie nach Möglichkeit die Anzahl und die Länge der Sprints in der agilen Entwicklungsphase.
In studentischen Projekten beschreibt die Roadmap den Arbeitsaufwand, der innerhalb der zugewiesenen Zeit geleistet wird. Eine gute Praxis in studentischen Arbeiten besteht darin, einen Meilenstein in der Mitte der Projektzeit einzubauen und die bis dahin erzielten Erfolge zu präsentieren.

Erstellen Sie einen Plan für die folgende Produktfindungsphase. Dieser Plan wird hauptsächlich durch die Wahl der Methode oder Methoden des "Product Findings" beeinflusst. Bei Verwendung iterativer Methoden: Erwägen Sie, Meilensteine als Entscheidungspunkte dafür zu verwenden, ob zusätzliche Iterationen erforderlich sind.

Identifizieren Sie die Risiken, bewerten Sie sie und definieren Sie gegebenenfalls Gegen- oder Minderungsmassnahmen. Wiederholen Sie den Risikomanagementprozess bei jeder Iteration.

Halten Sie alle Ergebnisse dieser Phase im Projektmanagementplan fest.

4.2. Produktfindung (product finding)

In der Product Finding Phase wird das "Initial Product Backlog" erstellt, eine Sammlung aller bisher bekannten Anforderungen.
In einem ersten Schritt werden die Anforderungen in einer Form zusammengefasst, die für jede am Projekt beteiligte Person verständlich ist (Epics). Epics werden verwendet, um sowohl funktionale als auch nicht-funktionale Anforderungen zu beschreiben.
Formulieren Sie für jede Anforderung mindestens einen aussagekräftigen Titel und eine kurze, adäquate Beschreibung. Ergänzen Sie anschliessend jede Anforderung mit einer Priorität und Komplexität: Das Team sollte zuerst die wertvollsten Anforderungen implementieren. Der Grad der Komplexität dient als erste Abschätzung des Implementierungsaufwands.

In einem ersten Ansatz kann eine einfache Tabelle für das Initial Product Backlog verwendet werden. Später kann ein Tool verwendet werden, das Revisionen unterstützt, da sich alle Einträge im Backlog im Laufe der Zeit verbessern oder ändern.

4.3. Evaluieren (evaluate)

Welche Technologien eignen sich für das Projekt? Nutzen Sie z.B. eine Nutzwertanalyse: Formulieren Sie für jede Technologieoption Kriterien, gewichten Sie jedes Kriterium und bewerten Sie dann jede Option. Es ist eine gute Praxis, Technologien zu verwenden, die dem Know-how und den Vorlieben des Teams entsprechen. Das Team kann nur dann gute Leistungen erbringen, wenn es mit der gewählten Technologie vertraut ist. In der anschliessenden Entwicklungsphase gibt es keinen Raum, um neue Technologien zu erlernen; es ist ausserdem äusserst schwierig, den Aufwand abzuschätzen, der erforderlich ist, um neue Fähigkeiten zu erwerben.

Dokumentieren Sie die Evaluierung im Softwarearchitektur Dokument.

4.4. Vorbereitung (prepare)

4.4.1.Vorbereitung - Projektplanung

Sobald das erste Product Backlog vorliegt, können weitere Planungsaktivitäten gestartet werden. Das Team hat nun eine Vorstellung vom Arbeitsumfang.
Gemeinsam mit dem Team kann der Projektleiter die Roadmap aktualisieren und den Release-Plan erstellen, einen lebendigen Plan, der für die nächsten Sprints (bis zum nächsten Release) zeigt, wann welches Backlog-Item implementiert wird.
Und da sich das Product Backlog im Laufe der Zeit ändert (neue Elemente, geänderte Priorität, ...), kann sich später auch der Release-Plan ändern. Daher kann der Release-Plan nach jedem Sprint überarbeitet werden.

Das Team aktualisiert die Risikoliste (und tut dies nach jedem Sprint).
Möglicherweise muss der Release-Plan aktualisiert werden, wenn präventive Massnahmen (z. B. die Erstellung eines Prototyps) zu zusätzlichen Backlog-Elementen (Spikes) führen.

Die Pläne und die aktualisierte Risikoliste werden in den Projektmanagementplan aufgenommen.

4.4.2.Vorbereitung – Infrastruktur

Richten Sie ein Projekt-Repository ein.
Bei der Verwendung von GitLab kann es vorkommen, dass der Inhalt des initialen Product Backlogs in das «Issues»-System übertragen wird [15].

Richten Sie eine Entwicklungsumgebung für jedes Teammitglied ein. Virtualisierung kann dazu beitragen, sicherzustellen, dass alle Teammitglieder mit denselben Versionen der Compiler, des Frameworks, der Bibliotheken, des Testframeworks usw. arbeiten.

Bereiten Sie die verbleibende Dokumentation vor und vereinbaren Sie, wie Sie bei der Aktualisierung zusammenarbeiten möchten. In der agilen Entwicklung ist es eine gute Praxis, die vollständige Dokumentation nach jedem Sprint auf dem neuesten Stand zu halten. Der Ansatz der lebenden Dokumentation, der auf grossen Plattformen wie GitLab oder GitHub unterstützt wird, gilt derzeit als bewährte Praxis. Diese Plattformen verwenden den Begriff «Seiten»

Die Dokumentation der Entwicklungsinfrastruktur ist Teil des Projektmanagementplans.

4.4.3.Vorbereitung – Requirements Engineering

Das Requirements Engineering in der Vorbereitungsphase umfasst die Verfeinerung der Backlogelemente, die im ersten Sprint ausgewählt werden können. Die Verfeinerung beginnt mit der Formulierung von User Stories mit den Akzeptanzkriterien. Story-Splitting wird für Backlogelemente verwendet, die zu gross sind. Definieren Sie anschliessend die detaillierten Anforderungen für jede User Story. Dies kann zum Beispiel mit Hilfe einer Anwendungsfallbeschreibung, einer Spezifikation durch ein Beispiel oder einem Storyboard für eine grafische Benutzeroberfläche erfolgen.
Die Vorbereitung des Requirements Engineering endet mit dem Backlog-Refinement-Meeting zwischen dem Team und dem Auftraggeber/Kunden. Dies ist notwendig, um dem Team die Möglichkeit zu geben, von Anfang an die richtige Software zu entwickeln.

4.4.4.Vorbereitung – Teststrategie

Wenn es an Zeit und Geld mangelt, liegt Testen nicht drin. Es ist deshalb wichtig, systematisch Testprioritäten mit einer passenden Teststrategie zu definieren.

Eine gute Teststrategie umfasst die folgenden drei Schritte:
1. Testfokus
2. Testpriorisierung
3. Grobe Festlegung der Tests und Testarten

Als Artefakt dokumentiert und legitimiert die Teststrategie diese drei Punkte:
1. Festlegen Testfokus:
   Je nach Produkt oder Projekt werden die folgenden Fragen beantwortet:
   - Welche rechtlichen Rahmenbedingungen sind von Belang (Bestimmungen, Datenschutz, Sicherheit, ...)?
   - In welcher Liga befinden wir uns?
   - Was ist die Lebensdauer des Produkts (Lebenszyklus)?
   - Was sind die möglichen Auswirkungen eines Fehlers oder Schadens (gefährdete Personen, Umwelt, Reputation, etc.)?
   - Welche Ressourcen stehen uns zur Verfügung?
2. Priorisierung:
   Angemessene Priorisierung stellt sicher, dass das nützlichste Item als erstes getestet wird. Die Strategie bestimmt, wie die Priorisierung zu geschehen hat.
   Risikobasiertes Testen (also Priorisierung anhand von Risiken) hat sich in der Vergangenheit bewährt. Die «RPI-Methode» bietet sich hier als Tool an.
3. Tests, Testarten:
   Basierend auf den in Schritt 1 und 2 gewonnenen Erkenntnissen werden in Schritt 3 die nötigen Tests und Testarten als Richtlinie definiert. Der «agile testing quadrant» [16]und die Qualitätskriterien gemäss ISO 9126 [17] sind hier nützliche Hilfsmittel.

Besonderes Augenmerk ist auf die allenfalls nötige Infrastruktur zu legen (s. auch Kapitel «Vorbereitung - Infrastruktur»), insbesondere für Last- und Sicherheitstests und Tests mit sensiblen Daten (Anonymisierung).
Diese Spezifikationen werden dann in der Teststrategie iterativ festgelegt.

4.5. Agile Entwicklung (agile development)

4.5.1.Agile Entwicklung – Sprints

Ein Sprint sollte nicht länger als zwei Wochen dauern, denn ein wesentlicher Erfolgsfaktor bei der agilen Entwicklung ist das unmittelbare und regelmässige Feedback.

Ein Sprint in der agilen Entwicklung beginnt mit einem Planungsmeeting. Das Team entscheidet, welche Backlogelemente implementiert werden. Das Team kann die Planungspoker-Technik verwenden, um den Aufwand abzuschätzen.
Für jedes ausgewählte Backlog-Element muss die Arbeit weiter in Aufgaben (Tasks) unterteilt werden, d. h. eine Arbeitseinheit, die eine Person innerhalb eines Tages bearbeiten kann. Typische Aufgaben sind: «Entwerfen der ...», «Datenbank vorbereiten», «Codieren der ...», «Test automatisieren», ...
Eine Aufgabe ist erst dann beendet, wenn die zugehörige Dokumentation aktualisiert wird!

Während des Sprints pflegt das Team kontinuierlich das Product Backlog. Die Verfeinerung der Backlog-Items für den nächsten Sprint ist verpflichtend. Halten Sie in der Mitte des Sprints das Backlog Refinement Meeting ab, um zu überprüfen, ob die detaillierten Anforderungen für den nächsten Sprint vom Team verstanden, vom Auftraggeber/Kunden akzeptiert und die Akzeptanzkriterien niedergeschrieben wurden.

Passend zum Projektfortschritt wächst und verändert sich auch der Testumfang mit jedem Sprint. Während in den ersten Sprints der Fokus auf der «Machbarkeit» und auf Funktionstests liegt, die beweisen, dass etwas funktioniert (positive Tests), müssen in den folgenden Sprints andere Aspekte betrachtet und gegebenenfalls geplant und durchgeführt werden.

Auch hier dienen die Risikobewertung und die Norm ISO 9126 (Softwarequalität) als Grundlage, um eine neue Perspektive einzunehmen und pragmatisch – oder agil, d.h. dem grössten Nutzen folgend – die Testabdeckung zu erweitern.
Folgende Punkte sind dabei besonders zu beachten:

- Konzeption und Aufbau von Negativtests
- Konzeption und Bau von Limit- und Ausnahmetests
- Erhöhte Automatisierung von Tests
- Pflegen, aktualisieren und erweitern Sie die notwendigen Testdaten, da diese oft anonymisiert werden müssen (Datenschutz) und immer relevant und im richtigen Format verfügbar gehalten werden müssen.
- Wartung, Anpassung und Erweiterung der Testinfrastruktur einschliesslich der notwendigen Berechtigungen und Lizenzen.

Auch diese Aktivitäten sollten in das Product Backlog aufgenommen werden, denn was nicht geplant ist, wird nie umgesetzt.

Am Ende jedes Sprints demonstriert das Team dem Kunden/Auftraggeber die laufende Software auf dem Live-System und bittet um Feedback. Bei Bedarf werden neue Backlog-Items eingefügt.

Das Sprint-Review-Meeting endet mit einer Vorschau auf den nächsten Sprint. Passt es in den Release-Plan oder gibt es einen Änderungswunsch? Und was ist mit den Risiken? Stehen weitere Massnahmen zur Risikominderung an?

Während der Sprint-Retrospektive identifiziert das Team Prozessverbesserungen. Es könnte z.B. erörtern, ob die Schätzungen während der Sprint-Planung korrekt waren und wenn nicht, wie sie verbessert werden können.

Geben Sie im Projektmanagementplan den geschätzten Aufwand und die tatsächlichen Kosten für jedes ausgewählte Backlogelement an. Die Erkenntnisse aus dem Sprint Review, der Retrospektive und eine aktualisierte Risikoliste ergänzen die Dokumentation des Sprint-Controllings im Rahmen des Projektmanagementplans.

4.6. DevOps

4.6.1.Definition

Bei einem DevOps-Modell sind Entwicklungs- und Betriebsteams nicht mehr getrennt.

Manchmal werden diese beiden Teams zu einem einzigen Team zusammengeführt, in dem die Ingenieure über den gesamten Anwendungslebenszyklus hinweg arbeiten. Die Arbeit reicht von der Entwicklung über den Test, die Bereitstellung bis hin zum Betrieb.

Das Spektrum der Fähigkeiten ist nicht auf eine einzige Funktion beschränkt.

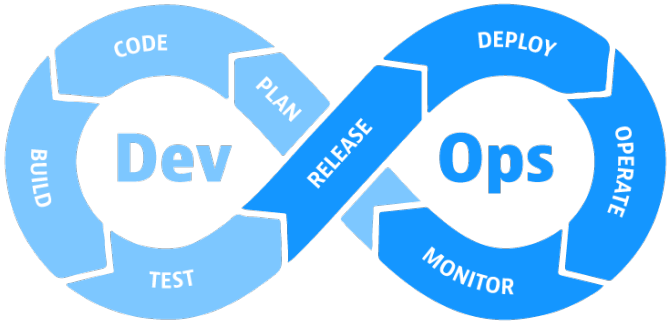


Abbildung 4: DevOps-Prinzip: Die Kombination von Entwicklung und Betrieb und den Prozessen, die ihn unterstützen, ermöglicht es Unternehmen, mit der Geschwindigkeit der Entwicklung Schritt zu halten.

In einigen DevOps-Modellen können Qualitätssicherungs- und Sicherheitsteams auch enger in die Entwicklung und den Betrieb integriert werden.

Wenn die Sicherheit im Mittelpunkt steht, wird dies manchmal auch als DevSecOps bezeichnet.

Diese Teams verwenden die Automatisierung von Prozessen, die in der Vergangenheit manuell und langsam abgearbeitet wurden. Sie verwenden einen Technologie-Stack und Tools, die ihnen helfen, Anwendungen schnell und zuverlässig zu betreiben und weiterzuentwickeln. Diese Tools helfen Ingenieuren auch dabei, selbstständig Aufgaben auszuführen, die normalerweise die Hilfe anderer Teams erfordert hätten. Dies erhöht die Geschwindigkeit eines Teams.

Wikipedia (Englisch) [18] enthält die folgende 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. Aktivitäten

DevOps-Teams üben täglich:

- **Coding**: Code-Entwicklung und -Überprüfung, Quellcode-Management-Tools, Code-Zusammenführung. Betonen Sie die kontinuierliche Entwicklung und das Testen. Kontinuierliches Testen kann nur vollständig automatisiert erfolgen.
- **Building**: Continuous Integration-Tools, Buildstatus. Schwerpunkt auf Continuous Integration, Continuous Delivery und Continuous Deployment
- **Testing**: Tools für kontinuierliche Tests, die schnelles und zeitnahes Feedback zu Geschäftsrisiken liefern. Unterstützt kontinuierliche Entwicklungsaktivitäten.
- **Packaging**: Artefaktrepository, Staging der Anwendung vor der Bereitstellung. Unterstützt die Continuous-Delivery-Aktivitäten in einem Unternehmensumfeld.
- **Releasing**: Änderungsmanagement, Freigaben von Freigaben, Automatisierung von Freigaben. Unterstützt die kontinuierliche Auslieferung und Bereitstellung von produktiven Servern.
- **Configuring**: Konfiguration und Verwaltung der Infrastruktur, Tools für Infrastruktur als Code. Unterstützt den Ansatz "Infrastruktur als Code".
- **Monitoring**: Überwachung der Anwendungsleistung, Endbenutzererfahrung. Betonen Sie die kontinuierliche Überwachung und Alarmierung und optimieren Sie die Zeit bis zur Wiederherstellung.

Hier sind einige der wichtigsten DevOps-Metriken:

- **Bereitstellungshäufigkeit**: Es wird analysiert, wie häufig Sie die aktuelle Version der Software in der Produktion bereitstellen. Die Bereitstellungsautomatisierung wird durch Continuous Deployment und Continuous Delivery abgedeckt. Höhere Frequenzen korrelieren mit Hochleistungsteams.
- Durchschnittliche Vorlaufzeit: Es gibt an, wie lange es dauert, eine brandneue Anforderung durch Vorlaufzeitverfolgung zu entwickeln, zu untersuchen, zu liefern und bereitzustellen. Wertstromansätze konzentrieren sich auf die Optimierung der Vorlaufzeit.
- **Meantime To Recovery**: Misst die Zeit zwischen einer Unterbrechung aufgrund einer Bereitstellung oder eines Systemausfalls und einer vollständigen Wiederherstellung durch die MTTR-Nachverfolgung (der mittleren Zeit bis zur Wiederherstellung). Der Fokus liegt auf einer effizienten Wiederherstellung und weg von der mittleren Zeit zwischen Ausfällen. Wenn Ihr Unternehmen innerhalb von Minuten eine Wiederherstellung durchführen kann, ist die Ausfallrate selten kritisch.
- **Änderungsfehlerrate**: Gibt an, wie oft die Änderungen oder Hotfixes eines Teams zu Fehlern führen, nachdem der Code bereitgestellt wurde.

Microsoft hat eine "DevOps-Checkliste" veröffentlicht. [19] um Ihre DevOps-Kultur und -Prozesse zu bewerten.

4.7. Abschliessen (finalize)

Um ein Projekt abzuschliessen, müssen zunächst alle verbleibenden Aufgaben erledigt werden. Es folgt die Übergabe der Ergebnisse an den Kunden oder die Stakeholder mit einer abschliessenden Freigabe. Abschliessend wird ein Projektabschlussbericht erstellt, in dem das Projekt bewertet und wichtige Erkenntnisse für zukünftige Projekte erfasst werden.

5. Artefakte in agile@HSLU

Wie im vorigen Kapitel erläutert, müssen alle gesammelten Informationen und alle Entscheidungen in dem einen oder anderen Artefakt niedergeschrieben werden. Die folgenden Erläuterungen zu den Inhalten typischer Artefakte in der Softwareentwicklung sollen als Leitfaden dienen.

5.1. Product Backlog

In der Phase der Produktfindung kann ein anfängliches Product Backlog durch eine einfache Tabelle dargestellt werden.

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						

Abbildung 5: Tabelle, die für das anfängliche Product Backlog verwendet wird

Später sollte ein Tool wie GitLab oder Jira [20] verwendet werden, um den Fortschritt über die Backlogelemente nachzuverfolgen. Die nächste Abbildung zeigt ein GitLab-Issue-Board (Quelle: GitLab Docs [21]).

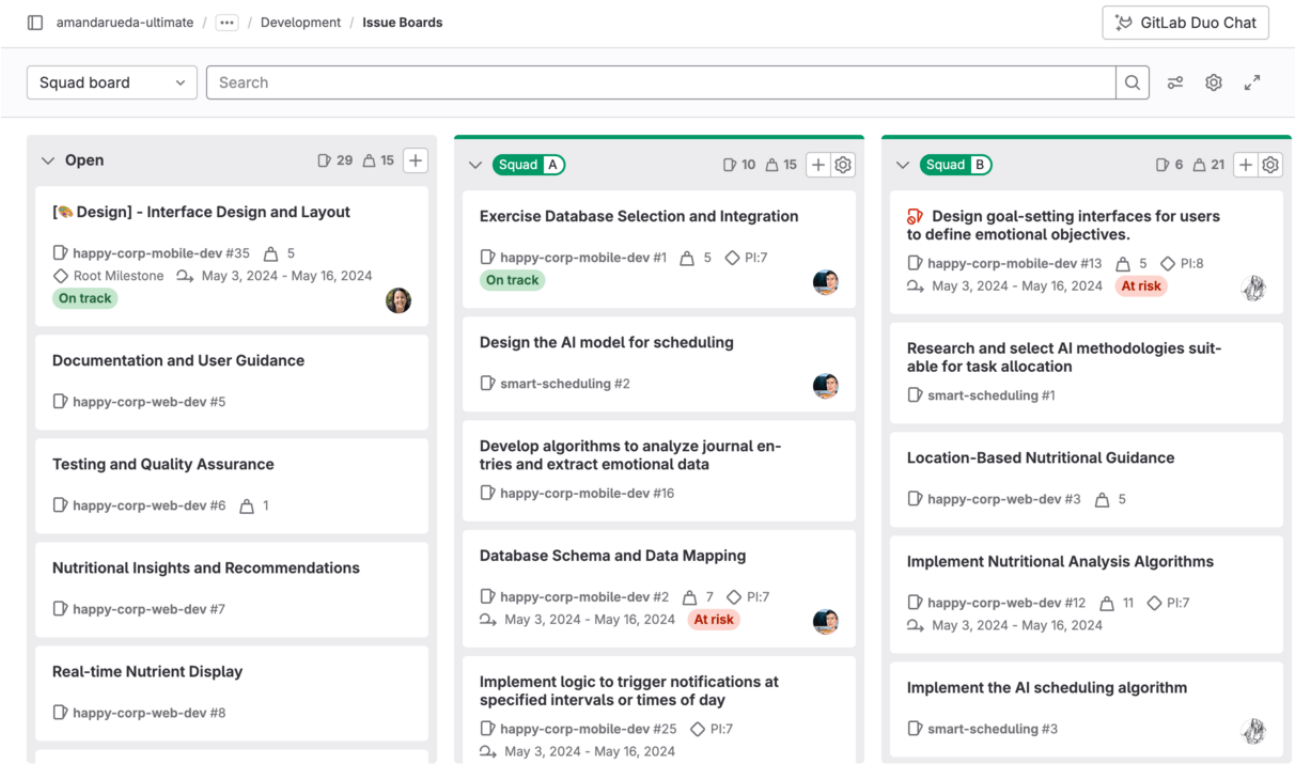


Abbildung 6: Beispiel für ein GitLab Issue Board

Für Angehörige der HSLU gibt es auf ILIAS eine Schulung zur Nutzung von GitLab [22].

5.2. Projektmanagement-Plan

Der Projektmanagementplan enthält alle Informationen und Entscheidungen, die den Entwicklungsprozess betreffen. Einige Inhalte des Plans bleiben über die gesamte Projektzeit konstant.
Andere Teile müssen nach jeder Iteration gepflegt und aktualisiert werden, wie z.B. der Release-Plan, die Risikoliste und das Projekt-Burndown-Chart, andere Teile sind pro Sprint einzigartig (z.B. Sprint-Planung und -Controlling, Review und Retrospektive Ergebnisse).

In den nächsten Unterkapiteln wird der Inhalt eines Good-Practice-Projektmanagementplans beschrieben.

5.2.1. Vision und Ziele

Die Projektvision dient als Inspiration und Fokussierung für das Projekt. Es artikuliert den Grund für das Projekt in kurzer und verständlicher Form, und es ist wichtig, dass das gesamte Team die gleiche Vision versteht, teilt und während des gesamten Projekts darauf hinarbeitet.
Ein Ziel ist ein gewünschter Zukunftszustand, der inhaltlich, zeitlich und umfangreich genau definiert ist. Versuchen Sie gemeinsam mit den Kunden und Auftraggeber, mindestens drei, aber nicht mehr als sechs Ziele zu formulieren.
Ziele sollten so "SMART" wie möglich formuliert werden. "SMART" steht für:

- **Spezifisch** Das Ziel muss für konkret benannte Organisationen, Rahmenbedingungen, etc. formuliert werden. Die Grenzen des Zielbereichs werden festgelegt.
- **Messbar** Qualitäten und ggf. Quantität der Zielerreichung werden ermittelt und zusammen mit Indikatoren abgeleitet.
- **Angemessen** Ist es das "richtige" Ziel? Gibt es einen Bedarf für die geplanten Massnahmen?
- **Realistisch** Die Aussichten, das Ziel zu erreichen, sind unter den gegebenen Rahmenbedingungen (Ressourcen, Zeit, Kompetenzen) hinreichend hoch; externe, unkontrollierbare Faktoren stehen der Zielerreichung nicht im Weg.
- **Terminiert** Es wird ein Zeitrahmen angegeben.

5.2.2. Umfang

Erstellen Sie eine Liste aller Artefakte, die Kunden und Auftraggeber als Ergebnis vom Projektteam erwarten.
In einem Softwareentwicklungsprojekt ist das Hauptergebnis die Software selbst. Kunden und Auftraggeber möchten jedoch möglicherweise Dokumente erhalten, die das Produkt beschreiben, zeigen, wie die Software getestet wurde, ihnen bei der Verwendung der Software helfen, ...
Liefergegenstände können sein:

- Anwendung
- Dokumentation der Softwarearchitektur
- Dokumentation zur Teststrategie und zum Testskript
- Benutzerhandbuch
- Bereitstellungs-, Installations- und Wartungshandbücher
- Projektmanagement-Plan

Kunden und Auftraggeber wünschen sich oft Zwischenergebnisse, um Teile des neuen Produkts so schnell wie möglich nutzen zu können (siehe Kapitel Roadmap).
Die folgende Tabelle kann verwendet werden, um die Projektergebnisse zu definieren:

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	

Abbildung 7: Liste der Lieferungen

Mit der Liste der Lieferungen definieren Sie eindeutig, was in den Umfang des Projekts fällt. Es empfiehlt sich, auch Teile, die "ausserhalb des Geltungsbereichs" liegen, klar zu definieren.
Vervollständigen Sie die Beschreibung des Umfangs, indem Sie mögliche Einschränkungen aufschreiben.

5.2.3.Projektorganisation

Erstellen Sie eine Liste mit den Personen, die an dem Projekt beteiligt oder vom Projekt betroffen sind (die Stakeholder), und beschreiben Sie deren Rolle.
Richten Sie das Projektteam ein und legen Sie fest, wer die Rolle des Projektleiters, des Scrum Product Owners und des Scrum Masters übernimmt.

5.2.4.Projektbegleitung

Beschreiben Sie (z.B. in Form einer Installationsanleitung), wie die Entwicklungsumgebung eingerichtet wird. Welche Tools, Server, Datenbanken, Frameworks und Bibliotheken werden für die Entwicklung und das Testen verwendet?
Skizzieren Sie die Organisation des Repositorys.

5.2.5.Definition of Ready, Definition of Done

Definieren Sie neben den Lieferbestandteilen Qualitätsaspekte, die das Team bei der Implementierung eines Backlog-Elements erfüllt.

Mit der Definition of Ready (DoR) formuliert das Team Kriterien für die Backlog-Refinement-Aktivitäten: Welche Details sind erforderlich, bevor ein Backlog-Item bereit ist, im nächsten Sprint implementiert zu werden.

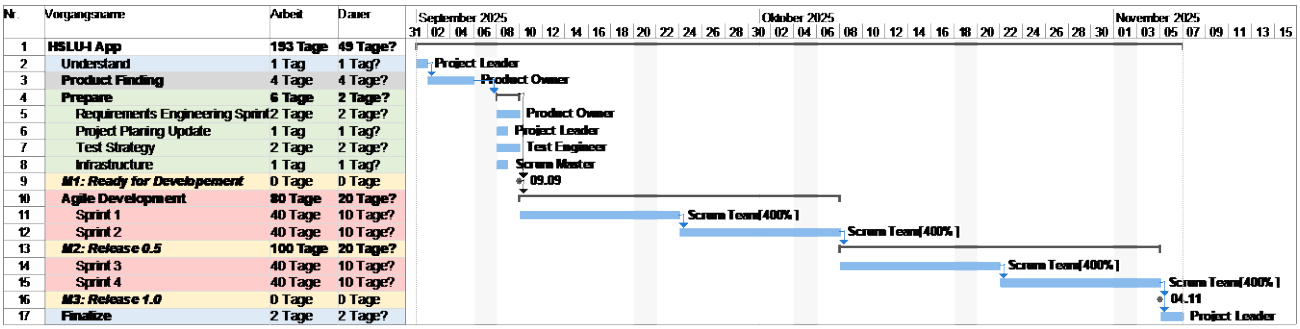
In der Beschreibung der Definition of Done (DoD) wird angegeben, wann ein Product Backlog-Element abgeschlossen ist. Dieses DoD ist ein Vertrag zwischen den Entwicklern und Kunden und stellt sicher, dass jeder im Team genau weiss, was erwartet wird. Es sorgt für Transparenz und Qualität, die für die Zwecke des Produkts und der Organisation geeignet ist.
In der Software kann eine Definition of Done lauten: "Done bedeutet nach Standards codiert, überprüft, implementiert mit Unit Test-Driven Development (TDD) getestet mit 100-prozentiger Testautomatisierung (oder gemäss der Teststrategie), integriert und dokumentiert."

5.2.6.Roadmap

Die Roadmap ist die langfristige Planung des Projekts und dient der Überprüfung der Wirtschaftlichkeit. Zentrale Elemente der Roadmap sind Phasen, Meilensteine und die Sprints in der Phase "Agile Development".

Ein Meilenstein ist ein geplanter Punkt im Projektablauf, an dem vordefinierte, messbare [Zwischen-] Ergebnisse zur Verfügung stehen, die es erlauben, den Projektablauf und die Projektkosten zu ermitteln. Legen Sie für jeden Meilenstein fest, welche Lieferobjekte verfügbar sein sollen. Der Meilenstein ist erreicht, wenn die erforderlichen Lieferobjekte verfügbar sind und ihre Überprüfung erfolgreich war.

Eine Roadmap mit drei Meilensteinen kann so aussehen.



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

Abbildung 8: Roadmap mit Phasen, Meilensteinen und Sprints

5.2.7.Release-Plan

Der Release-Plan in der agilen Entwicklung ist der kurzfristige Plan und gibt an, was nach den nächsten Sprints verfügbar ist. Er basiert auf den priorisierten Einträgen des eigentlichen Product Backlogs und zeigt auf, welche Backlog Items in welchem der folgenden Sprints umgesetzt werden.
Der Release-Plan kann eine einfache Tabelle oder mehrere Issue Boards in GitLab sein.

Titel	Description	Priority	Complexity	Estimated Effort	Sprint objectives
Free text	*Free text*	*1 - high* *2 - medium* *3 - low*	*1 - low* *2 - medium* *3 - high*	*amount of work* *in person days*	*Free text*
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	

Abbildung 9: Beispiel für einen Release-Plan als einfache Tabelle

Der Release-Plan ist ein lebendiger Plan. Das bedeutet, dass nach jedem Sprint der Release-Plan aktualisiert wird und der Inhalt des nächsten Sprints hinzugefügt wird. Bewahren Sie für jede Iteration einen Schnappschuss des Release-Plans auf.

5.2.8. Risikomanagement

Der Zweck des Risikomanagements besteht darin, potenzielle Probleme zu erkennen, bevor sie auftreten, und dann während des gesamten Projektlebenszyklus Massnahmen zu ergreifen und Messungen vorzunehmen, um sicherzustellen, dass diese Probleme nicht auftreten und die Projektziele gefährden.
Um Risiken zu identifizieren, verwenden Sie Checklisten, führen Sie ein Brainstorming mit Stakeholdern durch und nutzen Sie Erfahrungen aus früheren Projekten.
Beschreiben Sie jedes Risiko und bewerten Sie die Eintrittswahrscheinlichkeit und die Auswirkungen. Für die höchsten Risiken (siehe Risikomatrix) identifizieren Sie Indikatoren für das Eintreten und erarbeiten Sie vorbeugende Massnahmen, um die Wahrscheinlichkeit zu mindern und/oder Korrekturmassnahmen zur Verringerung des daraus resultierenden Schadens. Versuchen Sie als nächstes, die Wirkung der Massnahmen abzuschätzen.
Um all diese Informationen zu sammeln, kann die folgende Tabelle verwendet werden.

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													

Abbildung 10: Tabelle, die für das Risikomanagement verwendet wird.

Um die tatsächliche Risikosituation und die Auswirkungen Ihrer Risikomanagementmassnahmen zu visualisieren, verwenden Sie Risikomatrizen.

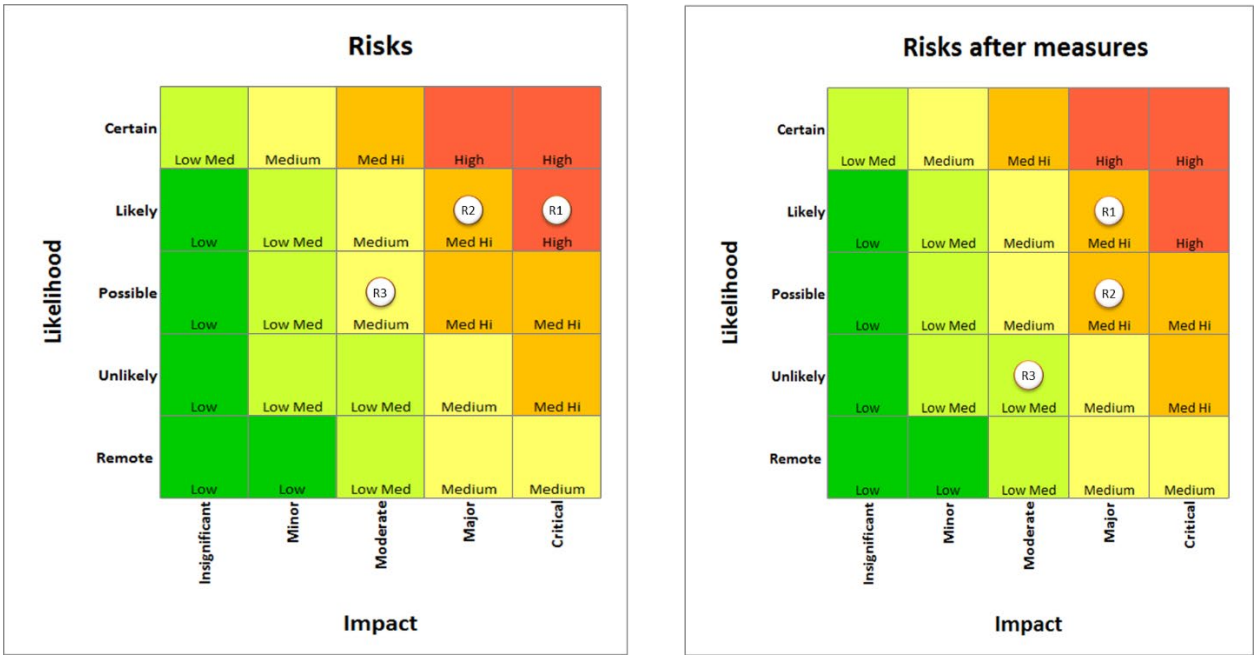


Abbildung 11: Risiko-Matrizen

Es ist wichtig zu prüfen, ob die im Risikomanagement erfassten und beschriebenen Risiken mit den in der Teststrategie genannten übereinstimmen. Ziel ist es, sicherzustellen, dass "risikobasiertes Testen" mit den Projektrisiken übereinstimmt.

Bewahren Sie für jede Iteration einen Schnappschuss der Risikoliste auf.

5.2.9.Controlling

Die Projektsteuerung umfasst alle Tätigkeiten, um projektbedingte Abweichungen zwischen Plan- und Ist-Zustand zu erkennen.
Gründe für Abweichungen können sein:

- Geänderte oder neue Anforderungen erhöhen den Projektaufwand.
- Fehler bei der Aufwandsschätzung in der Sprint-Planung
- Falsch verstandene Anforderungen
- Auftreten von Risiken
- Ineffiziente Teamarbeit

Um die oben genannten Abweichungen zu erkennen, gibt es für jeden Grund ein Werkzeug.

Geänderte oder neue Anforderungen
Das **Projekt Burndown Chart** zeigt die verbleibende Arbeitsmenge, die im Product Backlog im Zeitverlauf enthalten ist. Die Sprints werden als Zeitticks auf der x-Achse verwendet.
Die Trendlinie auf dem Projekt Burndown Chart tendiert in der Regel nach unten. Wenn dem Backlog jedoch neue Elemente hinzugefügt werden, kann die verbleibende Gesamtpunktzahl steigen.

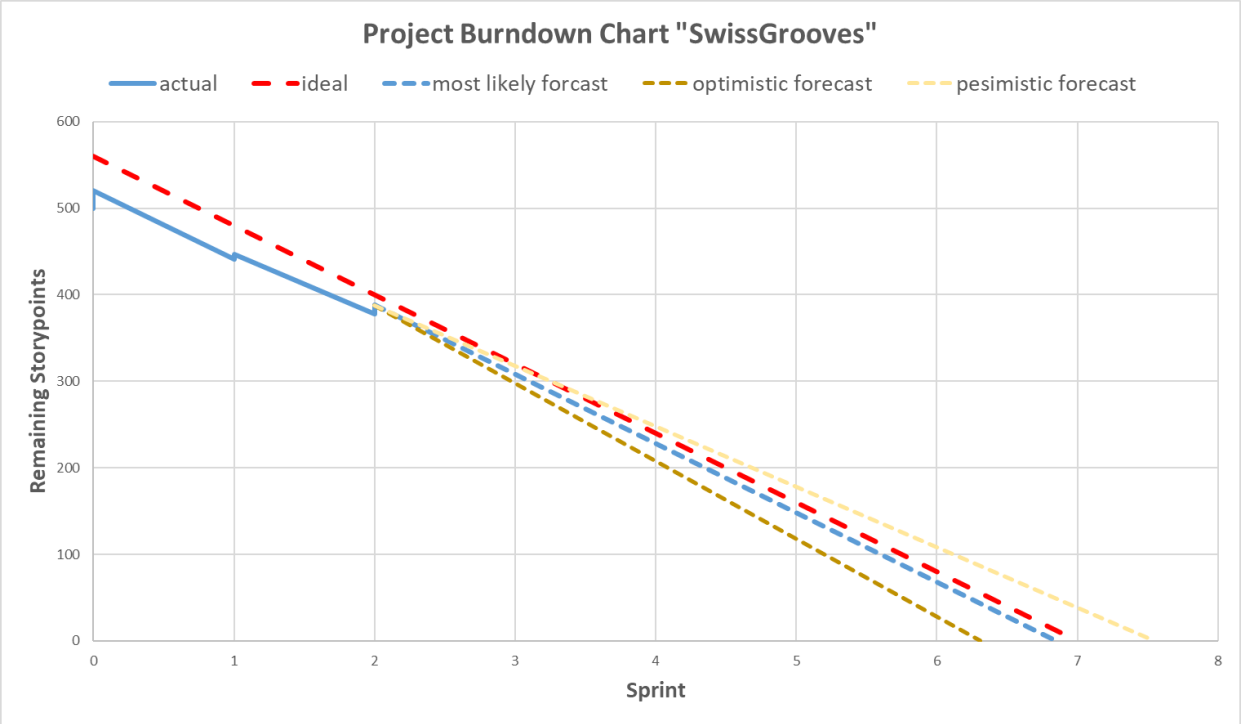


Abbildung 12: Projekt-Burndown-Diagramm

In der Literatur wird das Projekt Burndown Chart oft auch als Release-Burndown-Chart bezeichnet. Der Begriff "Release" wird dort verwendet, um die Lieferungen für einen Meilenstein zu umschreiben.

Fehler bei der Aufwandsschätzung in der Sprint-Planung
Um die Fähigkeit des Teams bei der Aufwandsschätzung in der Sprintplanung zu verbessern, verwenden Sie für jeden Sprint eine einfache Tabelle, um **die Schätzungen mit dem tatsächlichen Aufwand zu vergleichen**. Natürlich ist es notwendig, dass jedes Teammitglied die geleistete Arbeit pro Aufgabe oder User Story notiert, um echte Ist-Werte zu erhalten.

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

Abbildung 13: Rückblick auf die Sprint- Leistungen

Diese Ergebnisse im Team widerspiegeln (→ Retrospektive)

Falsch verstandene Anforderungen
Am Ende eines Sprints präsentiert das Team das Ergebnis im Rahmen des **Sprint-Reviews** den Stakeholdern. Das Product Backlog kann auch angepasst werden, um neuen Möglichkeiten gerecht zu werden. Berichten Sie für jeden Sprint über die Ergebnisse des Sprint-Reviews.

Auftreten von Risiken
Im Rahmen des Projektcontrollings werden die Risiken periodisch überwacht. Bei Bedarf wird erneut eine Risikoanalyse durchgeführt, z.B. bei neuen Backlog Items oder grösseren Änderungen im Projekt. Halten Sie für jeden Sprint nachvollziehbar alle **Änderungen in der Projektrisikoliste** fest.

Ineffiziente Teamarbeit
Der Zweck der **Sprint-Retrospektive** besteht darin, zu überprüfen, wie der vergangene Sprint in Bezug auf die beteiligten Personen, Beziehungen, Prozesse und Tools verlaufen ist. Berichten Sie für jeden Sprint über die Ergebnisse der Retrospektive.

5.3. Dokumentation zur Softwarearchitektur

Das Dokument zur Softwarearchitektur beschreibt alle Aspekte des Produkts.
Eine gute Struktur dafür wird im Rahmen der Initiative arc42 von Gernot Starke, Peter Hruschka und Ralf D. Müller beschrieben [23].
Einen Überblick über die Kapitelstruktur des arc42 Software Architecture Dokuments finden Sie in der nächsten Abbildung.

Abbildung 14: Aufbau der Dokumentation zur Softwarearchitektur nach arc42

Auf der arc42-Website wird https://arc42.org/ detailliert erläutert, wie die Softwarearchitektur aufgebaut, kommuniziert und dokumentiert wird.
Neben einer detaillierten Erläuterung des zu erwartenden Inhalts der verschiedenen Kapitel enthält die Website Tipps und Beispiele aus der Praxis.

6. Verzeichnisse

6.1. Bibliographie

[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. Liste der Abbildungen