Luis
Lüscher

Autor Luis Lüscher
Datum 28. September 2020
Version 0.1
Klassifikation Öffentlich
Seiten 22, inkl. Deckblatt

Vorgehensmodelle

Modul 306 IT-Kleinprojekt abwickeln

Änderungsverzeichnis

Version	Status	Name	Datum	Beschreibung
0.1	Erledigt	Lüscher, Luis	28.09.2020	Dokument wurde erstellt

Lizenz

Creative Commons License

Sie dürfen:

Teilen - das Material in jedwedem Format oder Medium vervielfältigen und weiterverbreiten

Bearbeiten – das Material remixen, verändern und darauf aufbauen

Unter folgenden Bedingungen:

Namensnennung – Sie müssen angemessene Urheber- und Rechteangaben machen, einen Link zur Lizenz beifügen und angeben, ob Änderungen vorgenommen wurden. Diese Angaben dürfen in jeder angemessenen Art und Weise gemacht werden, allerdings nicht so, dass der Eindruck entsteht, der Lizenzgeber unterstützt gerade Sie oder Ihre Nutzung besonders.

Nicht kommerziell – Sie dürfen das Material nicht für kommerzielle Zwecke nutzen.

Weitergabe unter gleichen Bedingungen – Wenn Sie das Material remixen, verändern oder anderweitig direkt darauf aufbauen, dürfen Sie Ihre Beiträge nur unter derselben Lizenz wie das Original verbreiten.

Keine weiteren Einschränkungen – Sie dürfen keine zusätzliche Klauseln oder technische Verfahren einsetzen, die anderen rechtlich untersagen, was die Lizenz erlaubt.

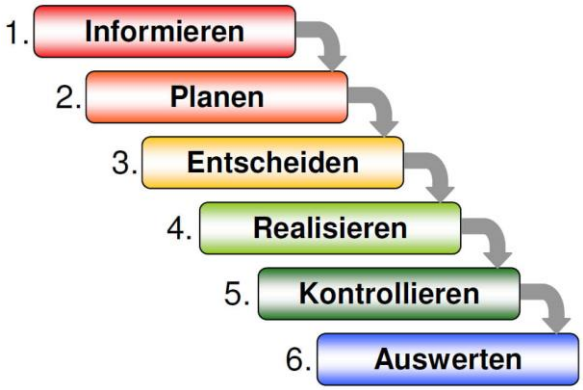
Inhaltsverzeichnis

1. Phasenmodelle (klassisch)

Es gibt unterschiedliche Phasenmodelle (standardisierte Projektstrukturen für die Erstellung des Projektproduktes), je nachdem, welche Art Projektprodukt erstellt wird.

1.1. IPERKA

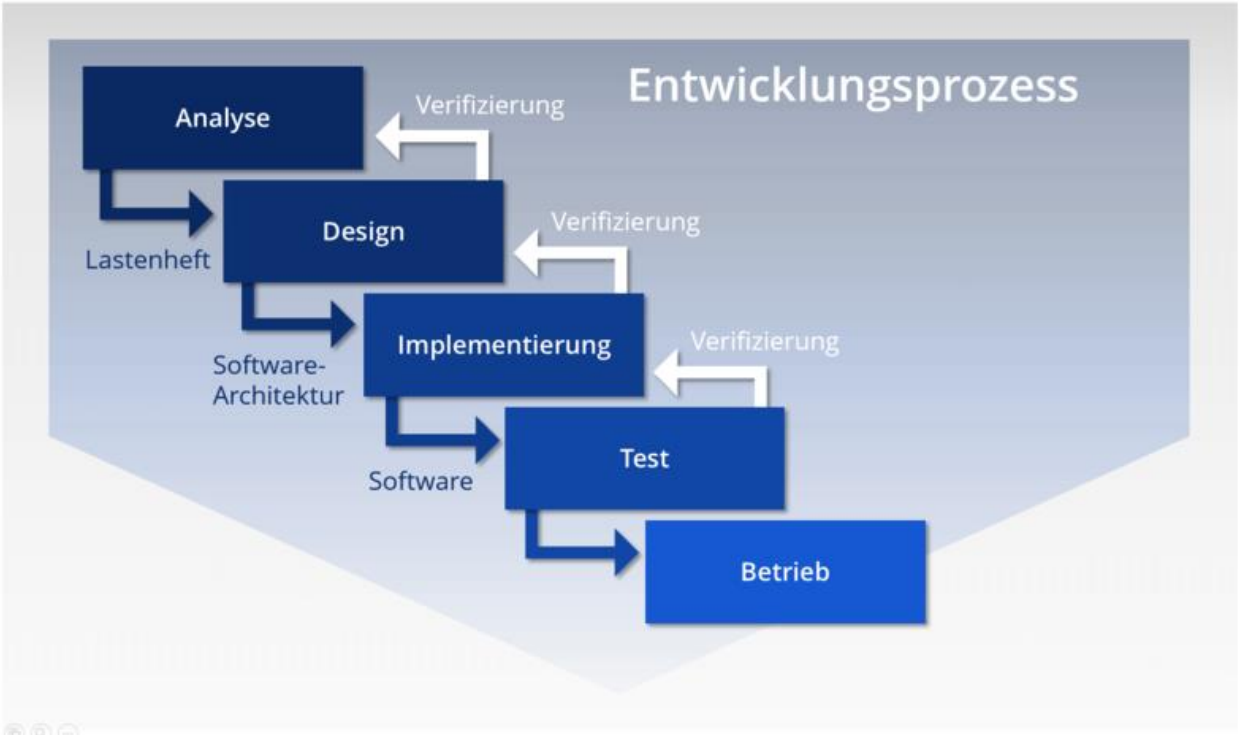
Bekanntes Phasenmodell für kleine IT-projekte. Bekannt aus dem Modul 431. Nicht geeignet für SW-Projekte



Informieren	• Auftrag klären • Informationen beschaffen • Informationen sortieren, ordnen, werten • Wesentliches erkennen
Planen	• Ziel definieren • Lösungsweg bestimmen • Arbeitsplan erstellen • Zeitplanung vornehmen • Ausschlusskriterien bezüglich Machbarkeitsanspruch
Entscheiden	• Point of no return • Strategie festlegen • Verbindlichkeiten absprechen und festhalten • Nutzwertanalyse von verschiedenen Varianten • Argumentarium erstellen und prüfen
Realisieren	• Ziel-Ausrichtung überprüfen • Probleme beheben • Zwischenziele vornehmen • Irrwege erkennen • Evtl. Entscheid für/ gegen Abbruch
Kontrollieren	• Meilenstein überprüfen • Vergleich von Planung und Umsetzung • Checkliste, eigene und Fremdkontrolle • Qualitätskontrolle • Abnahmekriterien überprüfen
Auswerten	• Reflexion über Produkt • Reflexion über Prozess • Reflexion über Zusammenarbeit, Umgang miteinander • Optimierung formulieren • Erkenntnisse zusammenfassen

1.2. Wasserfall

In der Praxis kommen verschiedene Versionen des Wasserfallmodells zum Einsatz. Gängig sind Modelle, die Entwicklungsprozesse in fünf Phasen unterteilen. Die von Royce definierten Phasen 1, 2 und 3 werden mitunter als Anforderungsanalyse in einer Projektphase zusammengefasst.

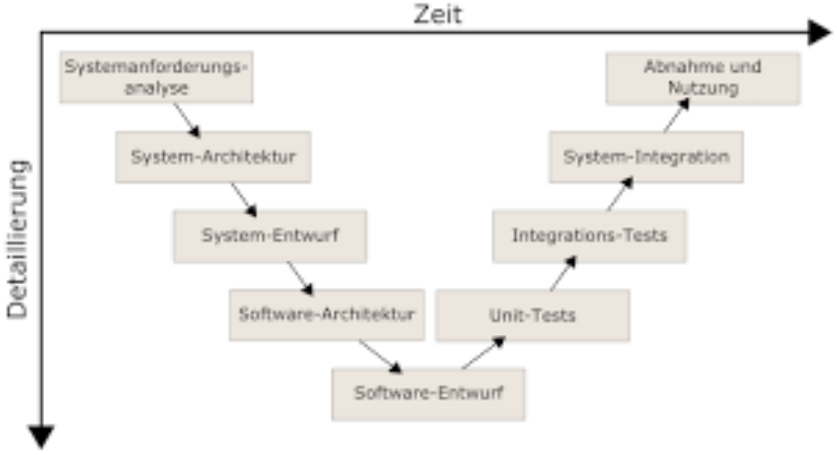


1.2.1. Die Phasen des Wasserfallmodells

Analyse: Planung, Anforderungsanalyse und -spezifikation.
Design: Systemdesign und -spezifikation
Implementierung: Programmierung und Modultest
Test: Systemintegration, System – und Integrationstests
Betrieb: Auslieferung, Wartung, Verbesserung

1.3. V-Modell XT

Das V-Modell XT definiert einige Grundkonzepte bzw. Begriffe, die für das Verständnis und die Anwendung wesentlich sind:



- **Tailoring**

  Da das V-Modell XT in vielen Konstellationen anwendbar ist, muss es zu Projektbeginn auf das konkrete Projekt hin zugeschnitten werden. Dies wird als Tailoring bezeichnet. Der Zusatz „XT“ beim V-Modell steht für „eXtreme Tailoring“ und betont die Anpassbarkeit des Modells an spezifische Projektsituationen.
- **Projekttypen**

  Das V-Modell XT klassifiziert Projekte mittels sogenannter Projekttypen und die Rolle, die eine Partei in einem Projekt einnimmt (also Auftraggeber, Auftragnehmer oder beides innerhalb einer Organisation).
- **Projekttypvarianten**

  In den Projekttypvarianten werden die Rahmenbedingungen für den Ablauf des Projekts festgehalten. So wird unterschieden, ob es sich um ein Auftraggeber- oder Auftragnehmerprojekt handelt, ob es einen oder mehrere Auftragnehmer gibt oder ob ein neues Produkt entwickelt oder ein bestehendes gepflegt werden soll.
- **Produkte und Aktivitäten**

  Die zu erarbeitenden Ergebnisse und Zwischenergebnisse werden im V-Modell XT als Produkte bezeichnet. Produkte werden von Mitarbeitern mit definierten Rollen und durch genau festgelegte Aktivitäten erzeugt. Ein Produkt kann sich in verschiedene Themen gliedern; auch für diese Themen kann es Aktivitäten geben. Produkte und Aktivitäten lassen sich zu Disziplinen zusammenfassen.
- **Vorgehensbausteine**

  Das V-Modell XT gilt als reichhaltiges, umfassendes Vorgehensmodell. Damit die Planung von Projekten besser gelingt, definiert es
  Vorgehensbausteine. Ein Vorgehensbaustein kapselt Aktivitäten, Produkte und Rollen.
- **Projektdurchführungsstrategien**

  Vorgehensbausteine machen keinerlei Angaben zur Reihenfolge der Durchführung des Vorhabens – sie ergibt sich aus der Projektdurchführungsstrategie und aus der zeitlichen Anordnung der Entscheidungspunkte. Die Projektdurchführungsstrategie wird durch die Projekttypvariante und die Festlegung von Projektmerkmalen bestimmt.
- **Entscheidungspunkt**

  Ein Entscheidungspunkt ist ein Meilenstein in einem V-Modell XT Projekt, an dem eine Entscheidung über die Fortführung des Projekts und die Freigabe der nächsten Projektphase entschieden wird. Für das Passieren eines Entscheidungspunkts muss ein zugeordnetes Produkt in einer definierten Qualität vorliegen.

1.4. HERMES

1.4.1. Überblick

- HERMES ist ein Kunstwort, das für den ursprünglichen Nutzen des Modells steht: **H**andbuch der **e**lektronischen **R**echenzentren des Bundes, **M**ethode für die **E**ntwicklung von **S**ystemen.
- Offener Standard zur Führung und Abwicklung von Projekten. Ist in der Schweiz bei öffentlichen IKT Projekten verbindlich vorgegeben (in Ausschreibungen wird oft ein zertifizierter Hermes Projektmanager gefordert).
- HERMES ist der schweizerische Projektmanagementstandard für Informations- und Kommunikationstechnikprojekte des Bundes und wird mittlerweile auch in kantonalen Verwaltungen eingesetzt

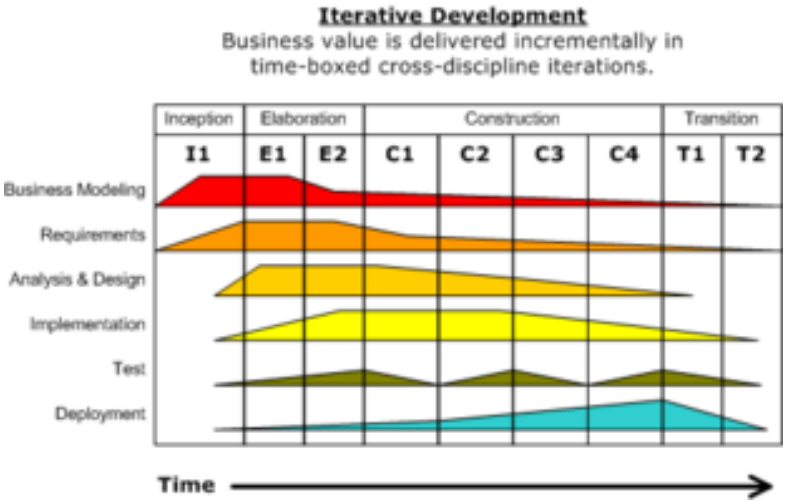
1.4.2. Ziele

- Gute Qualität der Informatiksysteme
- Verbesserte Kommunikation zwischen Fachabteilung/Anwendern und Informatikbereich
- Verkleinerte Projektrisiken
- Reduzierter Entwicklungsaufwand
- Hohe Transparenz bei der Spezifikation von Projektarbeiten

1.4.3. Eigenschaften

- HERMES gehört zu den linearen Vorgehensmodellen. Der Ablauf ist in sechs Phasen gegliedert. Diese sind im Einzelnen wie folgt bezeichnet: Initialisierung, Voranalyse, Konzept, Realisierung, Einführung und Abschluss. Am Ende jeder Phase steht ein Phasen–Entscheidungspunkt an dem die produzierten Ergebnisse verifiziert werden. Hier wird entschieden ob in die nächste Phase übergegangen werden kann, oder ob die vorherige Phase wiederholt werden muss.
- Bei der Durchführung der Projektphasen werden drei Aspekte unterschieden: `Wie?' ist das konkrete Vorgehen? `Was?' wird als Ergebnis der Phase erwartet? `Wer?' übernimmt welche Aufgaben bei der Durchführung?
- Das V-ModellXT und HERMES sind sich sehr ähnlich.
  Beide sind Phasenmodelle und definieren erwartete Ergebnisse pro Phase, legen beteiligte und verantwortliche Rollen fest und planen die Erstellung der Ergebnisse mit Aufgaben bzw. Aktivitäten. […] Die Trennung von Organisations- und Projektrollen ist bei beiden Vorgehen vorhanden.
- Es werden zwei Projekttypen unterschieden: Systementwicklung (make) und Systemadaption (buy)

1.5. Rational Unified Process (RUP)



1.5.1. Grundkonzepte und Phasen

RUP deckt weite Teile des Softwarelebenszyklus ab. Er unterscheidet zwischen einzelnen Lebenszyklusabschnitten (Phasen) inkl. der Iterationen in den einzelnen Abschnitten. Orthogonal dazu sind die Disziplinen positioniert, die in einem Projekt nach RUP die Aktivitäten bündeln. Zusammen ergeben sie das bekannte RUP-Gebirge (siehe [Kruchten 2003]). Im Kern folgt der RUP den folgenden drei Grundprinzipien:

- RUP ist Anwendungsfall-getrieben.
- Die Architektur steht im Zentrum der Planung.
- Das Vorgehen zur Entwicklungszeit ist iterativ/inkrementell.

1.5.2. Phasen und Meilensteine

Abschnitte im Lebenszyklus eines Projekts bezeichnet RUP als Phasen. RUP untergliedert ein Projekt in die folgenden vier Phasen:

- Inception (Konzeptionsphase)
- Elaboration (Entwurfsphase)
- Construction (Konstruktionsphase)
- Transition (Übergabephase)

In jeder der vier Phasen werden Aktivitäten (Arbeitsschritte) einzelner Disziplinen gebündelt. Weiterhin sind die einzelnen Phasen in Iterationen unterteilt. Jede Iteration schließt mit einem Meilenstein (Business Decision Point) ab:

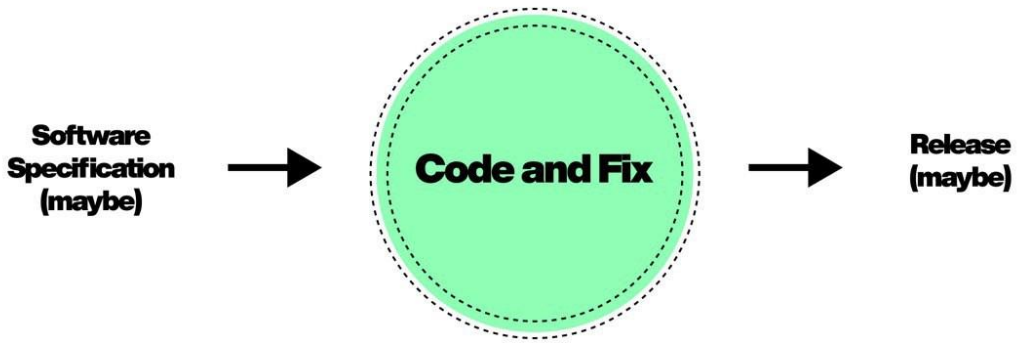
- Lifecycle Objectives
- Lifecycle Architecture
- Initial Operational Capability (Entwurfsmodelle und Beta-Release der Software)
- Product Release

Im ersten Meilenstein (lifecycle objectives) erwartet RUP die folgenden Ergebnisse: eine Projektvision inklusive eines rudimentären Anwendungsfallmodells, das die wesentliche Funktionalität des Zielsystems beschreibt. Zum Einsatz kommen hier üblicherweise UML-Anwendungsfalldiagramme. Weiterhin ist bereits ein erster, grober Architekturentwurf (z.B. mithilfe von UML-Komponenten- oder Klassendiagrammen) zu liefern. Organisatorisch soll hier bereits eine erste Identifikation relevanter Projektrisiken vorliegen.
Im zweiten Meilenstein (lifecycle architecture) ist ein Architekturprototyp vorzulegen. Weiterhin soll das initiale Anwendungsfallmodell weiter verfeinert werden. Organisatorisch ist zu diesem Meilenstein eine detaillierte Arbeitsplanung für die folgende Konstruktionsphase vorzulegen. Aus der Architektur sind daher bereits Arbeitspakete abzuleiten.
Der dritte Meilenstein (initial operational capability) fordert detaillierte Entwurfsmodelle (in verschiedenen UML-Diagrammtypen) und eine Beta-Version der Software. Während im vierten Meilenstein (product release) bereits eine produktive Software vorliegen soll.

2. Iterative Vorgehensmodelle (Agil)

Wenn schnelle, flexible Entwicklung gewünscht ist, eignen sich eher Vorgehensmodelle mit sich ab wechselnd wiederholenden Teilabschnitten:

2.1. Code & Fix



«Code and Fix» steht für «Programmierung und Fehlerbehebung», teilweise auch bekannt als «Build-and-Fix» Zyklus. Darunter wird ein völlig unstrukturiertes Vorgehen verstanden. Entstanden ist dieses Vorgehensmodell in der Anfangszeit der Rechentechnik, als Software meist von einem einzigen Entwickler produziert wurde.
Der Entwickler beginnt direkt mit der Implementierung des Systems. Die Entwicklung wird solange fortgesetzt, bis das System den Vorstellungen des Programmierers entspricht. Fehler werden beim Auftreten durch den Programmierer behoben. Dieser Ansatz ist unstrukturiert, es wird kaum Dokumentation erstellt und Analyse und Entwurf werden ebenfalls fast komplett ausgespart. Das

Vorgehensmodell «Code and Fix» ist dann erfolgsversprechend, wenn Entwickler und späterer Nutzer Rollen einer einzigen Person sind. Auch wenn dieses Vorgehensmodell durchaus qualitativ hochwertige Software hervorbringen kann, ist das Vorgehen in der professionellen Softwareentwicklung abzulehnen, da eine Wartung oder Weiterentwicklung der Software meist nur durch den ursprünglichen Entwickler möglich ist.

2.2. Prototyping

Der Begriff Prototyping bezeichnet die Entwicklung eines Musters oder Prototyps im Bereich Software-Engineering. Prototyping kann als schrittweise Annäherung an das fertige Endprodukt betrachtet werden: Ein Prototyp wird im Laufe des Projekts zu einem fertigen Produkt – zum Beispiel zu einer Website, einer App oder einer komplexeren Software-Anwendung. Protoyping ist eine Vorgehensweise, die sehr früh Feedback von beteiligten Entwicklern und vor allem von den Endanwendern ermöglicht, indem ein starkes Augenmerk auf die Kommunikation während des Entwicklungs-Prozesses gelegt wird.

Allgemein können Methoden im Bereich Prototyping danach unterschieden werden, ob sie vertikal oder horizontal angelegt sind.

- Horizontales Prototyping fokussiert sich auf einen speziellen Bereich einer Anwendungssoftware – zum Beispiel auf die Benutzerschnittstelle. Dabei fehlt der Bezug zu den technischen Funktionalitäten des Gesamtsystems und deren Implementation. Ziel ist es, den Nutzer oder Auftraggeber mit der GUI in Berührung zu bringen und erstes Feedback zu erhalten.

- Vertikales Prototyping greift sich einen speziellen Bereich einer Software heraus und zeigt die Wechselwirkungen mit anderen Komponenten des Systems. Eine Benutzerschnittstelle würde hier bereits mit der Datenverwaltung und anderen Teilen des Systems gemeinsam abgebildet werden. Das Ziel besteht darin, komplexe Funktionalitäten zu erläutern und die Software in Teilen durch den Anwender überprüfen zu lassen.

Darüber hinaus lassen sich Prototyping-Ansätze mithilfe von Anwendungszwecken voneinander trennen.

- Exploratives Protoyping: Das Anforderungsprofil einer Software wird Schritt für Schritt geklärt, indem Prototypen iterativ und schnell in einer Testumgebung erzeugt werden. Anschließend werden die Funktionalitäten verfeinert, um zu beurteilen, ob die Software das angenommene Problem löst und einen Nutzerbedarf abdeckt. In diesem Zusammenhang

wird auch von Demonstratoren, Rapid sowie Paper Prototyping gesprochen. Demonstratoren werden in der Akquise und in frühen Phasen eines Projekts verwendet, um abstrakte Anforderungen und Problemstellungen bei der Entwicklung deutlich zu machen und zu kommunizieren.

- Experimentelles Protoyping: Ein Entwurf wird mit den grundlegenden Funktionen erstellt und im Hinblick auf dessen Realisierbarkeit überprüft. Die gewonnenen Erkenntnisse dieses Experiments oder Tests fließen in das tatsächliche Produkt ein. Oft wird hier von Labormustern, Throw-Away-Protoypes gesprochen. Diese Muster sollen dabei helfen, technische Fragestellungen zu beantworten und das Projekt als solches auf dessen Realisierbarkeit hin überprüfen zu können.
- Evolutionäres Protoyping: Die Software wird sukzessive erstellt. Bei jedem Entwicklungsschritt sorgen Feedbackschleifen mit Nutzern, Entwicklern und Auftraggebern dafür, dass das Endprodukt das Anforderungsprofil erfüllt. In der Regel wird eine Version der Software stets lauffähig gehalten. Hier wird auch von Pilotsystemen gesprochen. Sie beinhalten bereits eine Vielzahl der anvisierten Funktionen und können von Anwendern sorgfältig geprüft werden.

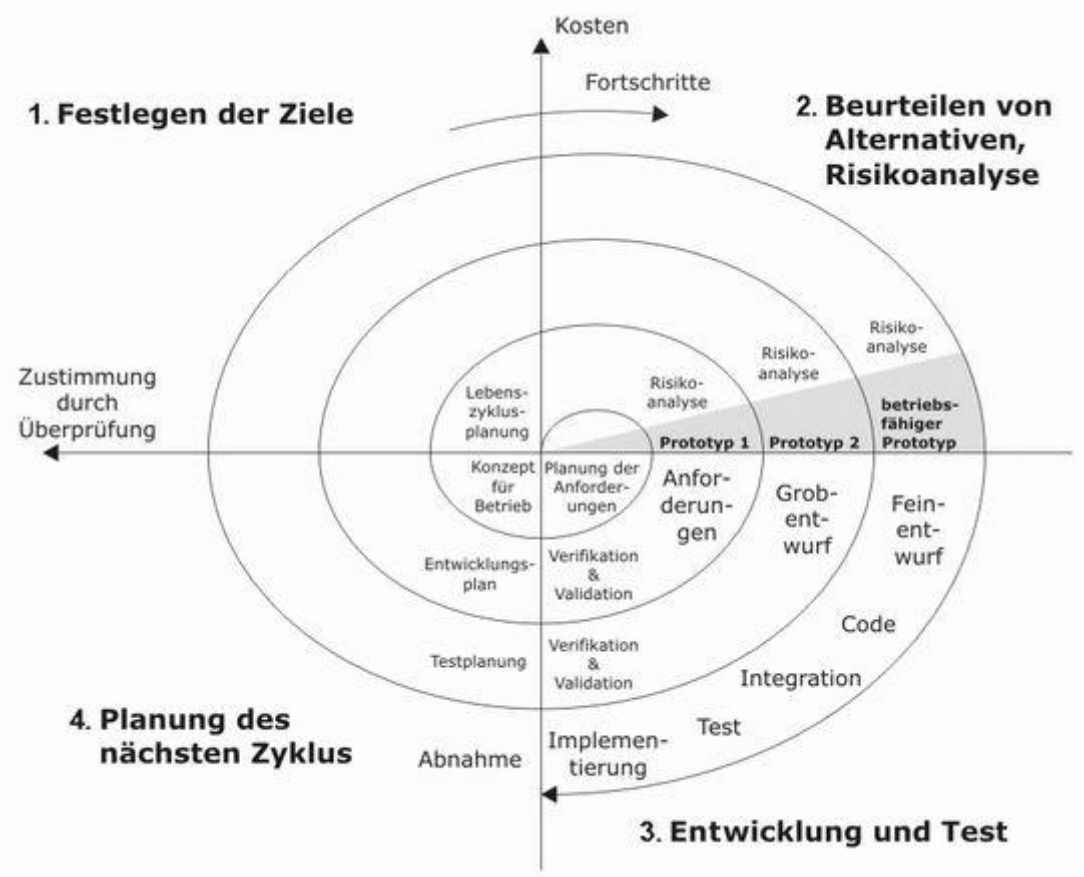
Die Wahl eines geeigneten Modells (vertikal, horizontal) und eines Anwendungszweckes ist im Einzelfall von vielen verschiedenen Faktoren abhängig. Das Budget, das Ziel des Projekts und die beteiligten Akteure (zum Beispiel externe Agenturen) bilden den Rahmen bei der Ausrichtung des Prototypings. In der Praxis können Modell und Anwendungszweck so gewählt werden, dass Mischformen der oben gemachten Unterscheidungen entstehen und einzelne Aspekte wie das Nutzer-Feedback besonders hervorgehoben werden.

2.3. Spiralmodell

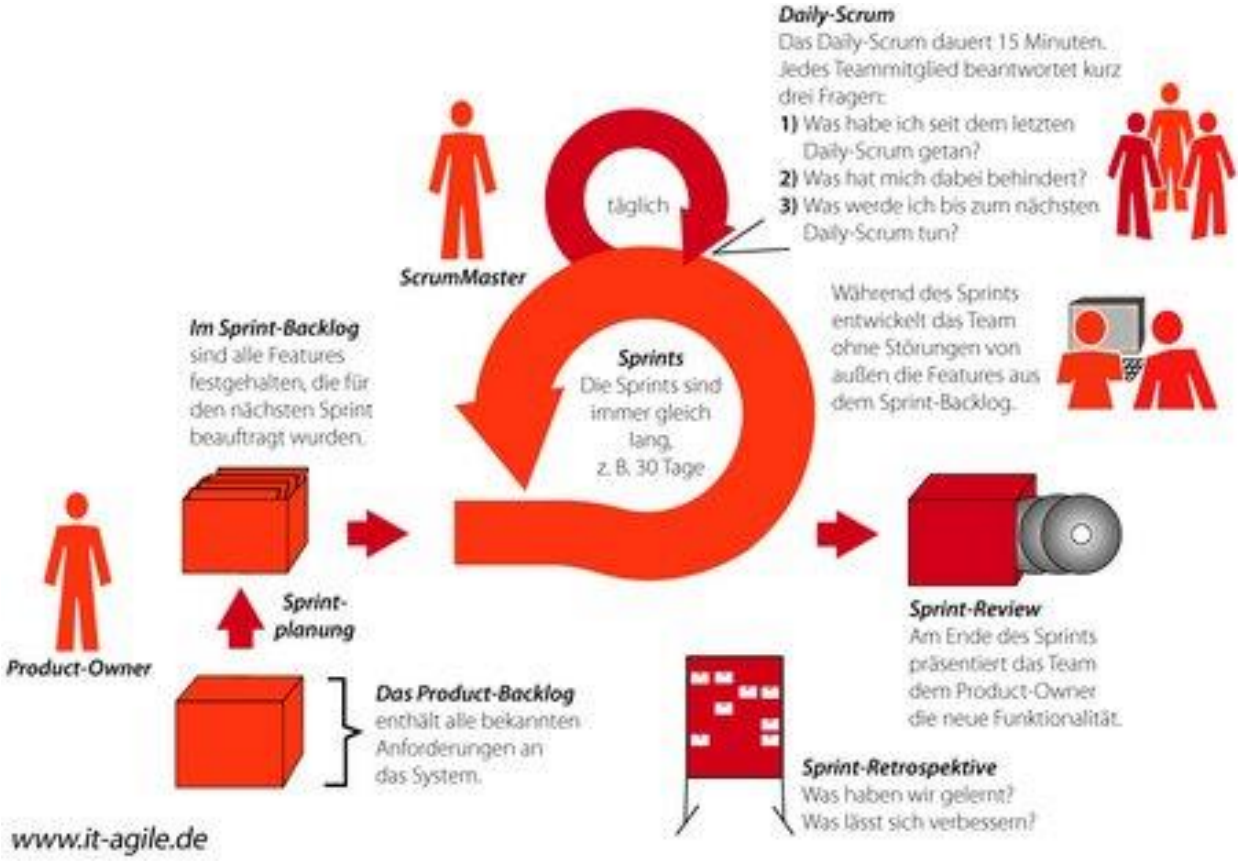
Das Spiralmodell kommt als risikoorientiertes Prozessmodell im Rahmen der Software-Entwicklung zum Einsatz. Es basiert auf dem Wasserfallmodell und kann andere Entwicklungsmodelle als Spezialfall enthalten.

Eine überarbeitete, detailliertere Version der Grafik mit der Struktur des Spiralmodells findet sich in einem Text aus dem Jahr 2000. Der Entwicklungsprozess durchläuft die Spirale von innen nach außen. Jede Umrundung stellt aufeinander aufbauende Phasen der Entwicklung dar. Ein Entwicklungsprozess beginnt im Inneren der Spirale mit Risikoanalyse, grundlegender Konzeptentwicklung und der Bestimmung von Anforderungen.

Der erste Quadrant bezieht sich jeweils auf die Ermittlung von Zielen, der zweite auf die Bewertung von Alternativen und die Reduktion von Risiken. Im dritten Quadranten erfolgt die eigentliche Entwicklung, im Softwarebereich speziell die Implementierung. Der vierte Quadrant leitet schließlich mit der Planung der nächsten Phasen wieder in den ersten über.



2.4. Agile Scrum



Im Mittelpunkt von Scrum steht das selbstorganisierte Entwicklerteam, das ohne Projektleiter auskommt. Der Scrum-Master sorgt als Methodenfachmann dafür, dass der Entwicklungsprozess nicht zerbricht. Dem Produktverantwortlichen (Product-Owner) kommt die Aufgabe zu, Anforderungen zu definieren, zu priorisieren und auch zu tauschen. Das Entwicklerteam arbeitet direkt mit dem Produktverantwortlichen zusammen.
Das Entwicklungsteam arbeitet in ungestörten Entwicklungszyklen von einer bis vier Wochen (Sprints). Der Produktverantwortliche kann innerhalb dieses Entwicklungszyklus keine Änderungen an den für diesen Zeitraum geplanten Anforderungen vornehmen, da dieser das Entwicklerteam in seiner Arbeit stören würde. Während eines Sprints wird der Product-Owner seine Vorstellungen von der weiteren Entwicklung in das Product Backlog aufnehmen und so für kommende Sprints vorzusehen.

2.5. Agile XP

Extreme Programming (XP) ist ein agiles Software-Entwicklungs-Framework, das darauf abzielt, qualitativ hochwertigere Software und eine höhere Lebensqualität für das Entwicklungsteam zu erreichen. XP ist das spezifischste der agilen Frameworks in Bezug auf geeignete technische Praktiken für die Softwareentwicklung.

2.5.1. Werte

Die fünf Werte des XP sind Kommunikation, Einfachheit, Feedback, Mut und Respekt und werden im Folgenden ausführlicher beschrieben.

2.5.2. Kommunikation

Software-Entwicklung ist von Natur aus ein Mannschaftssport, der auf Kommunikation angewiesen ist, um Wissen von einem Teammitglied zu allen anderen im Team zu übertragen. XP betont die Bedeutung der geeigneten Art der Kommunikation - persönliche Gespräche mit Hilfe eines Whiteboards oder eines anderen Zeichenmechanismus.

2.5.3. Einfachheit

Einfachheit bedeutet: "Was ist das Einfachste, was funktionieren wird? Der Zweck ist, Verschwendung zu vermeiden und nur die absolut notwendigen Dinge zu tun, wie z.B. das Design des Systems so einfach wie möglich zu halten, damit es leichter zu warten, zu unterstützen und zu überarbeiten ist. Einfachheit bedeutet auch, nur die Anforderungen anzusprechen, von denen Sie wissen; versuchen Sie nicht, die Zukunft vorherzusagen.

2.5.4. Rückmeldung

Durch ständiges Feedback über ihre bisherigen Bemühungen können die Teams Bereiche mit Verbesserungsbedarf identifizieren und ihre Praktiken überarbeiten. Das Feedback unterstützt auch eine einfache Gestaltung. Ihr Team baut etwas auf, sammelt Feedback zu Ihrem Entwurf und Ihrer Umsetzung und passt dann Ihr Produkt in Zukunft an.

2.5.5. Mut

Kent Beck definierte Mut als "wirksames Handeln im Angesicht der Angst" (Extreme Programming Explained S. 20). Diese Definition zeigt eine Präferenz für Aktionen, die auf anderen Prinzipien basieren, damit die Ergebnisse für das Team nicht schädlich sind. Man braucht Mut, um organisatorische Probleme anzusprechen, die die Effektivität des Teams beeinträchtigen. Sie brauchen Mut, damit aufzuhören, etwas zu tun, was nicht funktioniert, und etwas anderes auszuprobieren. Sie müssen den Mut haben, Feedback zu akzeptieren und danach zu handeln, auch wenn es schwer zu akzeptieren ist.

2.5.6. Respekt

Die Mitglieder Ihres Teams müssen sich gegenseitig respektieren, um miteinander zu kommunizieren, Feedback zu geben und zu akzeptieren, das Ihre Beziehung ehrt, und um gemeinsam einfache Entwürfe und Lösungen zu finden.

2.6. Kanban

Kanban ist eine Arbeitsmanagement-Methode, die aus dem Toyota Production System (TPS) entstanden ist. Ende der 1940er führte Toyota die „Just-in-Time“-Produktion ein. Der Ansatz basiert auf einem Pull-System. Die Produktion wird dabei an der Kundennachfrage ausgerichtet und nicht

wie bei Push-Systemen üblich auf bestimmte Mengen festgesetzt, die dann auf den Markt kommen.
Dieses einzigartige Produktionssystem legte den Grundstein für eine Lean-Produktion oder einfach Lean. Sein Hauptzweck ist die Minimierung von Aktivitäten, die zu Verlusten führen, ohne die Produktivität zu beeinträchtigen. Das Hauptziel ist es, ohne Zusatzkosten mehr Wert für die Kunden zu schaffen.

Aus dem Japanischen wird Kanban wörtlich als **Schild** oder **visuelles Signal übersetzt**. Das einfachste Kanban-Board hat drei Spalten – „Angefordert", „In Bearbeitung" und „Erledigt". Korrekt aufgebaut und benutzt dient die Tafel als Informationsknoten, der zeigt, wo die Engpässe sind und was den reibungslosen Ablauf des Workflows behindert.

2.6.1. Prinzip 1

Kanban erfordert keine bestimmte Konfiguration und kann über einen vorhandenen Workflow oder Prozess gelegt werden, um Probleme zu erkennen. So kann Kanban in jedem Unternehmen leicht eingeführt werden, weil für den Einstieg keine umfassenden Änderungen nötig sind.

2.6.2. Prinzip 2

Die Kanban-Methode wurde entwickelt, um mit minimalem Widerstand zu kontinuierlichen, inkrementellen und evolutionären Veränderungen des aktuellen Prozesses beizutragen. Generell werden umfassende Änderungen vermieden, da sie Angst oder Unsicherheit auslösen und so auf Widerstand stossen.

2.6.3. Prinzip 3

Kanban berücksichtigt den möglichen Wert vorhandener Prozesse, Rollen, Verantwortlichkeiten und Titel deren mögliche Beibehaltung. Die Kanban-Methode verbietet Änderungen nicht, schreibt sie aber auch nicht vor. Sie fördert inkrementelle Veränderungen, da sie nicht die Art von Angst auslöst, die den Fortschritt behindert.

2.6.4. Prinzip 4

Das ist das neueste Kanban-Prinzip. Einige der besten Führungsqualitäten ergeben sich aus den alltäglichen Maßnahmen von Menschen aus der Praxis. Nicht nur die Führungsebene, sondern alle müssen die kontinuierliche Verbesserung (Kaizen) für sich annehmen und umsetzen, damit auf im Team, in der Abteilung und firmenweit optimale Performance etabliert wird.

3. Vor- und Nachteile IPERKA und HERMES

3.1. Vor- und Nachteile IPERKA

Vorteile:

- Sie können sich die einzelnen Schritte leicht merken, da der Name IPERKA aus den Anfangsbuchstaben der zugehörigen Schritte besteht: Informieren, Planen, Entscheiden, Realisieren, Kontrollieren und Auswerten.
- Diese Methode fördert das methodische und strukturierte Vorgehen, da die einzelnen Tätigkeiten problemlos den entsprechenden Schritten zugeordnet werden können.
- Oft wird zuwenig geplant und zu früh realisiert. IPERKA legt starkes Gewicht auf die Planung. Erst wenn ein solides Konzept ausgearbeitet ist und die Entscheidung feststeht, welche Lösungsvariante umgesetzt wird, sollte mit der Realisierung begonnen werden.

- Oft wird davon ausgegangen, dass das notwendige Wissen bereits vorhanden ist. Zum Schritt "Informationen beschaffen" gehört z.B. auch, dass Sie sich alle notwendigen Kenntnisse über das erforderliche Ergebis (Produkt) aneignen.
- Auch eine Auswertung der Arbeit gehört zur strukturierten Vorgehensweise. Am Schluss schauen Sie auf die Realisierungsphase zurück, werten Ihre Erfahrungen aus und ziehen daraus Lehren für zukünftige Produkte ähnlicher Art.

Nachteile:

- Nicht geeignet für Software-Entwicklung

3.2. Vor- und Nachteile HERMES

Vorteile:

- Gute Qualität der Informatiksysteme
- Verkleinerte Projektrisiken
- Reduzierter Entwicklungsaufwand

Nachteile:

- Hohe administrative Aufwand während der Umsetzung und die Komplexität der Prozesse

3.2.1. Anwendungsbereiche Hermes

Geeignet: bei öffentlichen IKT Projekten verbindlich vorgegeben für Informations- und Komm-.technikprojekte des Bundes und wird mittlerweile auch in kantonalen Verwaltungen.

4. Vor- und Nachteile Wasserfall und RUP

4.1. Vor- und Nachteile RUP

Nachteile:

- hoher Managementaufwand (oft weitere Entscheidungen)
- Komplex & hoher Einarbeitungsaufwand
- hoher Aufwand für Iterationsplanung
- Werkzeugunterstützung herstellerabhängig (IBM)

Vorteile:

- für reine Softwareentwicklungsprojekte, bei denen der Kunde für Feedback und Entscheidungen zur Verfügung steht

4.1.1. Anwendugsbereiche

- Große und risikoreiche Projekte, insbesondere Use-Case-getriebene Entwicklung und schnelle Auslieferung hochwertiger Software.

4.2. Vor- und Nachteile Wasserfall

Vorteile:

- Einfache Struktur durch klar abgegrenzte Projektphasen.
- Gute Dokumentation des Entwicklungsprozesses durch klar definierte Meilensteine.
- Kosten und Arbeitsaufwand lassen sich bereits bei Projektbeginn abschätzen.
- Projekte die nach dem Wasserfallmodell strukturiert werden, lassen sich auf der Zeitachse gut abbilden.

Nachteile:

- Komplexe oder mehrschichtige Projekte lassen sich nur selten in klar abgegrenzte Projektphasen unterteilen.
- Geringer Spielraum für Anpassungen des Projektablaufs aufgrund veränderter Anforderungen.
- Der Endanwender wird erst nach der Programmierung in den Produktionsprozess eingebunden.
- Fehler werden mitunter erst am Ende des Entwicklungsprozesses erkannt.

4.2.1. Anwendungsbereich

- Einfache kleine oder mittelgroße Softwareprojekte mit klar definierten und unveränderlichen Anforderungen (Entwicklung einer Website für kleine Unternehmen).
- Projekte, bei denen eine strengere Kontrolle, vorhersehbare Budget und Zeitpläne erforderlich sind (z. B. staatliche Projekte).
- Projekte, bei denen mehrere Regeln und Vorschriften eingehalten werden müssen (medizinische Softwareprojekte).
- Projekte, bei denen ein bekanntes Technologie-Stack und Tools zum Einsatz kommen.

5. Vor- und Nachteil V-Modell und Spiralmodell und SCRUM

5.1. Vor- und Nachteile V-Modell XT

Es gibt unterschiedliche Meinungen zu den Vorteilen und Nachteilen des V-Modell XT. In der Einleitung der aktuellen Version werden folgende Vorteile genannt:

- Verbesserung der Kommunikation zwischen den Beteiligten
- Minimierung der Projektrisiken
- Gewährleistung von Qualität
- Eindämmung der Gesamtkosten über den gesamten Projekt- und Systemlebenszyklus
- Verbesserung der Informationssicherheit

Über jeden dieser Punkt lässt sich detailliert diskutieren. Genauso auch über manchmal genannte Nachteile:

- Komplexität und Umfang des Modells (die Dokumentation ist bspw. 213 Din-A4 Seiten lang)
- Parallelisierung von Aufgaben ist schwierig
- Änderungswünsche lassen sich nicht leicht integrieren
- Abgrenzung verschiedener Projektphasen funktioniert in der Praxis nicht
- Fokus auf Dokumentenerstellung ist zu gross

Und auch über jeden dieser Punkte können Meinungen divergieren.

5.1.1. Anwendungsbereiche

- Projekte, bei denen Störungen und Ausfallzeiten inakzeptabel sind (z. B. medizinische Software, Software für das Fuhrparkmanagement im Luftverkehr).

5.2. Vor- und Nachteile Spiralmodell

Vorteile:

- flexibles, generisches Modell
- frühe Einbindung von Auftraggebern und Anwendern möglich
- periodische, risikogetriebene Überprüfung
- perfekte Abstimmung zwischen technischen Anforderungen und Design
- maximale Kontrolle über Kosten, Ressourcen und Qualität des Softwareprojekts
- gut für neuartige technische Umgebungen geeignet

Nachteile:

- Hoher Managementaufwand
- Regelmässige Entscheidungen können den Entwicklungsprozess verzögern
- Durch den aufgegliderten entwicklungsprozess gelangen Fheler und konzeptionelle ungereimtheiten leicht in das Endprodukt
- Know-how in risikoanalyse bzw. Management essenziel, aber oft nicht vorhanden
- Für kleinerer Projekte mit überschaubarem Risiko ungeeignet.

5.2.1. Anwendungsbereiche

- Projekte mit unklaren Geschäftsanforderungen oder zu anspruchsvollen / innovativen Anforderungen.
- Bei großen und komplexen Projekten.
- Forschungs- und Entwicklungstätigkeit (F&E) oder Einführung einer neuen Dienstleistung oder eines neuen Produkts.

5.3. Vor- und Nachteile SCRUM

Vorteile von Scrum

- Wenige Regeln, leicht verständlich und schnell einführbar
- Kurze Kommunikationswege
- Hohe Flexibilität/Agilität durch adaptives Planen
- Hohe Effektivität durch Selbstorganisation
- Hohe Transparenz durch regelmäßige Meetings und Backlogs
- Zeitnahe Realisation neuer Produkteigenschaften bzw. Inkremente
- Kontinuierlicher Verbesserungsprozess
- Kurzfristige Problem-Identifikation
- Geringer Administrations- und Dokumentationsaufwand

Nachteile von Scrum

- Kein Gesamtüberblick über die komplette Projektstrecke
- Hoher Kommunikations- und Abstimmungsaufwand
- Wenige konkrete Handlungsempfehlungen
- Zeitverluste bei zu «defensiven» Sprintplanungen
- «Tunnelblick-Gefahr» bei ausschließlicher Fokussierung auf Tasks
- Erschwerte Koordination mehrerer Entwicklungsteams bei Großprojekten
- Potenzielle Verunsicherung aufgrund fehlender Zuständigkeiten und Hierarchien
- Potenzielle Unvereinbarkeit mit bestehenden Unternehmensstrukturen

6. SCRUM

Scrum ist eine agile Entwicklungsmethode. Die Entwicklung wird in Iterationen organisiert, die man in Scrum "Sprints" nennt. Eine gleichmäßige Sprintlänge gibt dabei dem Team einen Rhythmus. Ein Sprint dauert maximal vier Wochen. Damit werden frühe und regelmäßige Lieferungen erreicht. Am Anfang eines Sprints steht eine priorisierte Liste mit Anforderungen - das Product Backlog. Davon "zieht" sich das Entwicklungsteam entsprechend der Priorität so viele Anforderungen, wie es erfahrungsgemäß in einem Sprint umsetzen kann. Dann setzt das Team diese Anforderungen so um, dass am Ende vom Sprint wieder ein nutzbares Produkt existiert. Dieses zeigt das Team den Kunden, nimmt Feedback auf und passt damit die Anforderungen an. Zum Abschluss eines Sprints überlegt das Team, wie es seine Arbeitsweise verbessern kann. Dann beginnt der nächste Sprint.

Scrum hat ganz viele Vorteile, die am Ende zu mehr Nutzen bei weniger Aufwand führen - und das bei einem (oder gerade wegen einem) motivierten Team. Hier die wichtigsten Punkte:

- Durch die konsequente Priorisierung der Anforderungen auf Basis einer Kosten-Nutzen-Relation werden zuerst die Anforderungen umgesetzt, die einen hohen Nutzen bieten. Damit liegt schnell und früh ein nutzbares Ergebnis vor.
- Durch die regelmäßige Fertigstellung am Ende eines Sprints wird "Fertigwerden" und Produktqualität zur Routine - und nicht zum Drama.
- Frühe und regelmäßige Lieferungen reduzieren das Risiko des Scheiterns.
- Frühes Feedback eliminiert unnötige Anforderungen und damit Verschwendung.
- Die ständige Aktualisierung der Anforderungen macht Anforderungsänderungen einfach - und erlaubt den Kunden, das Produkt auf die Zielgerade zu lenken. So wird Nutzen statt Planerfüllung optimiert.
- Auf das Wesentliche reduzierte Planungstechniken senken die Hürde für das Projektmanagement und ermöglichen damit Aktualität und Disziplin.

Scrum definiert wenige Eckpfeiler beim Vorgehen - deshalb wird es häufig als Rahmenwerk (engl. Framework) bezeichnet. Im Kern stehen drei Rollen, drei Artefakte (Ergebnistypen) und vier Ereignisse.

Scrum-Rollen:

- Der Product Owner ist verantwortlich für die Anforderungen. Er priorisiert die Anforderungen und nimmt am Ende eines jeden Sprints die Umsetzung der Anforderungen ab.
- Der Scrum Master ist verantwortlich für die Sicherstellung, dass das Team operativ und produktiv arbeiten kann.
- Das Entwicklungsteam organisiert sich selbst und ist verantworlich für die Erreichung der Sprintziele.

Scrum-Ergebnisse:

- Im Sprint Planning werden für den Sprint die Anforderungen bestimmt, die im Sprint umgesetzt werden, und eine detaillierte Planung durchgeführt.
- Im Daily Scrum wird der Status des Sprints kurz besprochen. Jedes Teammitglied beantwortet die Fragen: „Was habe ich gestern getan?“, „Was tue ich heute?“ und „Was hindert mich?“
- Im Sprint Review geht es um die Vorstellung des „potentially shippable products“ und um das Feedback der Anwender, um das Produkt besser zu machen

- In der Sprint Retrospektive geht es um die Betrachtung der Arbeitsweise vom letzten Sprint und um Maßnahmen, um die Arbeit im kommenden Sprint zu verbessern.
- Das Product Backlog Refinement ist kein klassisches Scrum Ereignis, aber als kontinuierliche Aktivität ein wertvoller Vorgang . Hierzu gehört das Hinzufügen von Details, Schätzungen und einer Ordnung zu Einträgen im Product Backlog.

Scrum-Artifakte:

- Das Product Backlog ist eine Liste der Kundenanforderungen, die nach Wichtigkeit priorisiert sind.
- Das Sprint Backlog enthält die Aktivitäten, die durchgeführt werden müssen, um die Kundenanforderungen des Sprints umzusetzen.
- Das Produktinkrement ist das Teilprodukt, das am Ende eines Sprints erstellt wurde (alle bisherigen Funktionalität + die neuen Funktionalitäten)