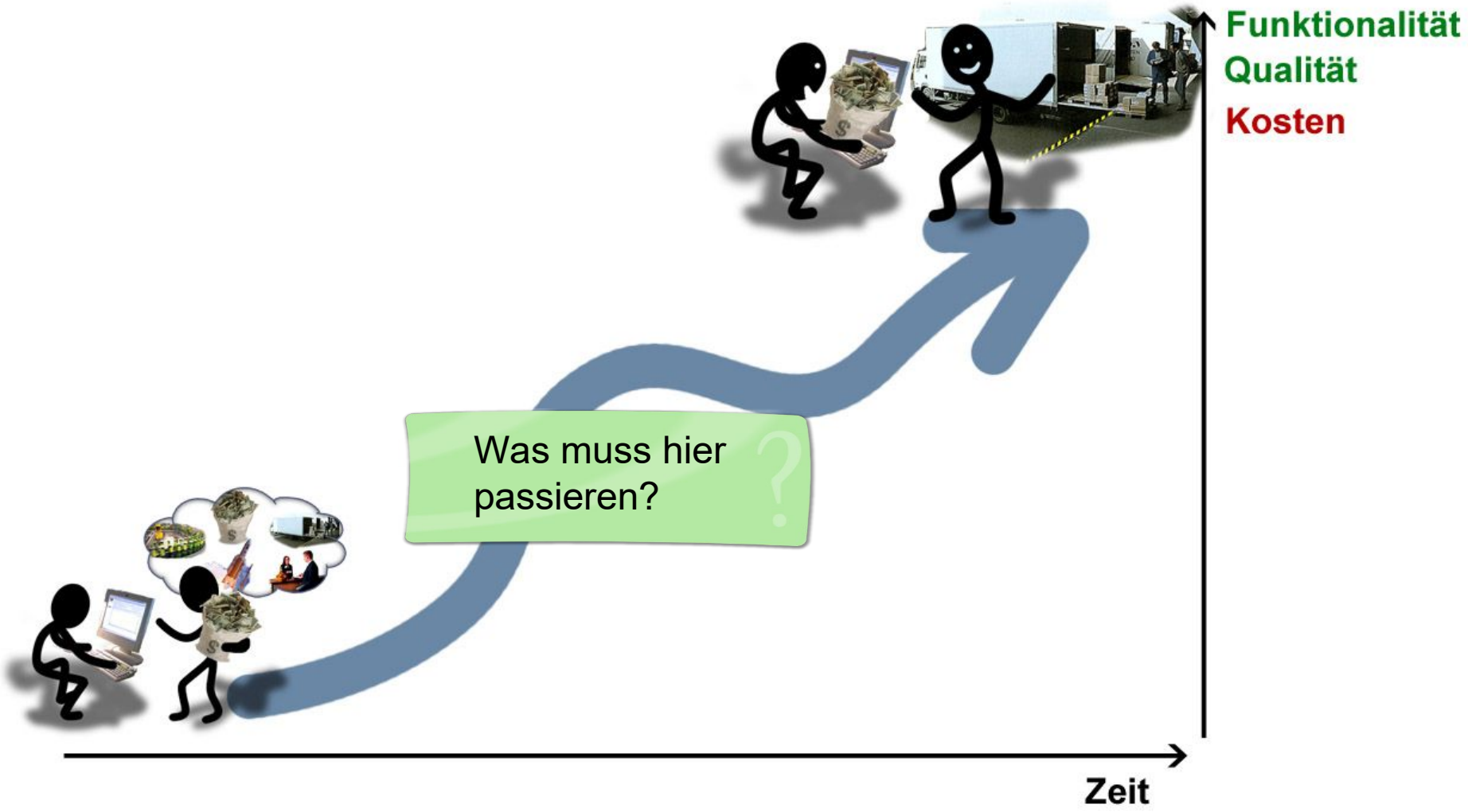
Software Engineering I

3. Vorgehensmodelle und Phasen

Prof. Dr. Eckhard Kruse

DHBW Mannheim

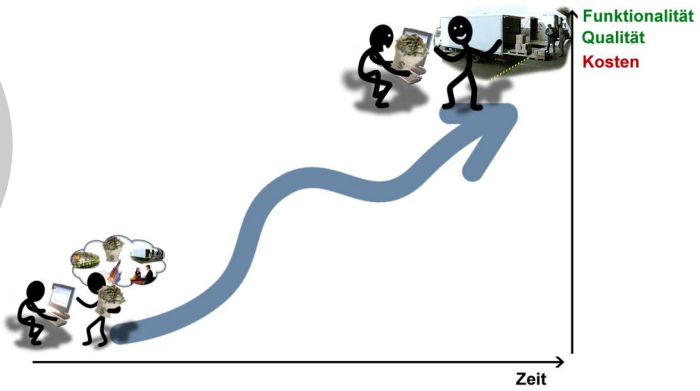
Vorgehensmodelle



„Code and fix“

Beispiel

1. Code schreiben
2. compilieren, testen, debuggen
3. Code verbessern
4. Nicht fertig? Gehe zu 2.



Reicht das?

- Keine dokumentierten Anforderungen / Interaktion mit dem Kunden?
- Keine Planung: Wann wird es fertig? Zu welchem Preis?
- Wartung? Dokumentation?
- …

→ “Hacker“-Ansatz: Bestenfalls für Mini-Projekte (und tolerante Kunden) geeignet.

Wasserfallmodell

Definition

Anforderungsanalyse

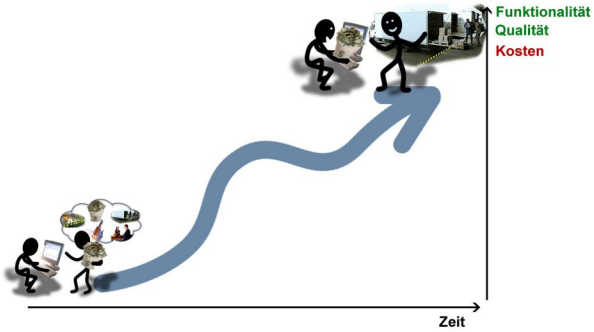
Entwurf

Implementierung

Test

Inbetriebnahme

Wartung



Hinweis:

Es gibt viele Wasserfall-Varianten bzgl. konkreter Anzahl und Namen der Phasen

Wasserfallmodell

Übung

Übung

3.1 Wasserfallmodell

Untersuchen Sie das Wasserfallmodell:

a) Recherchieren Sie im Internet nach dem Wasserfallmodell. Was für Information und Meinungen finden Sie dazu?

b) Nennen Sie Vor- und Nachteile des Modells (auch im Vergleich zu „Code und Fix“).

c) Können Sie sich Verbesserungen für dieses Modell vorstellen?

Wasserfallmodell - Bewertung

Übung

Gut:

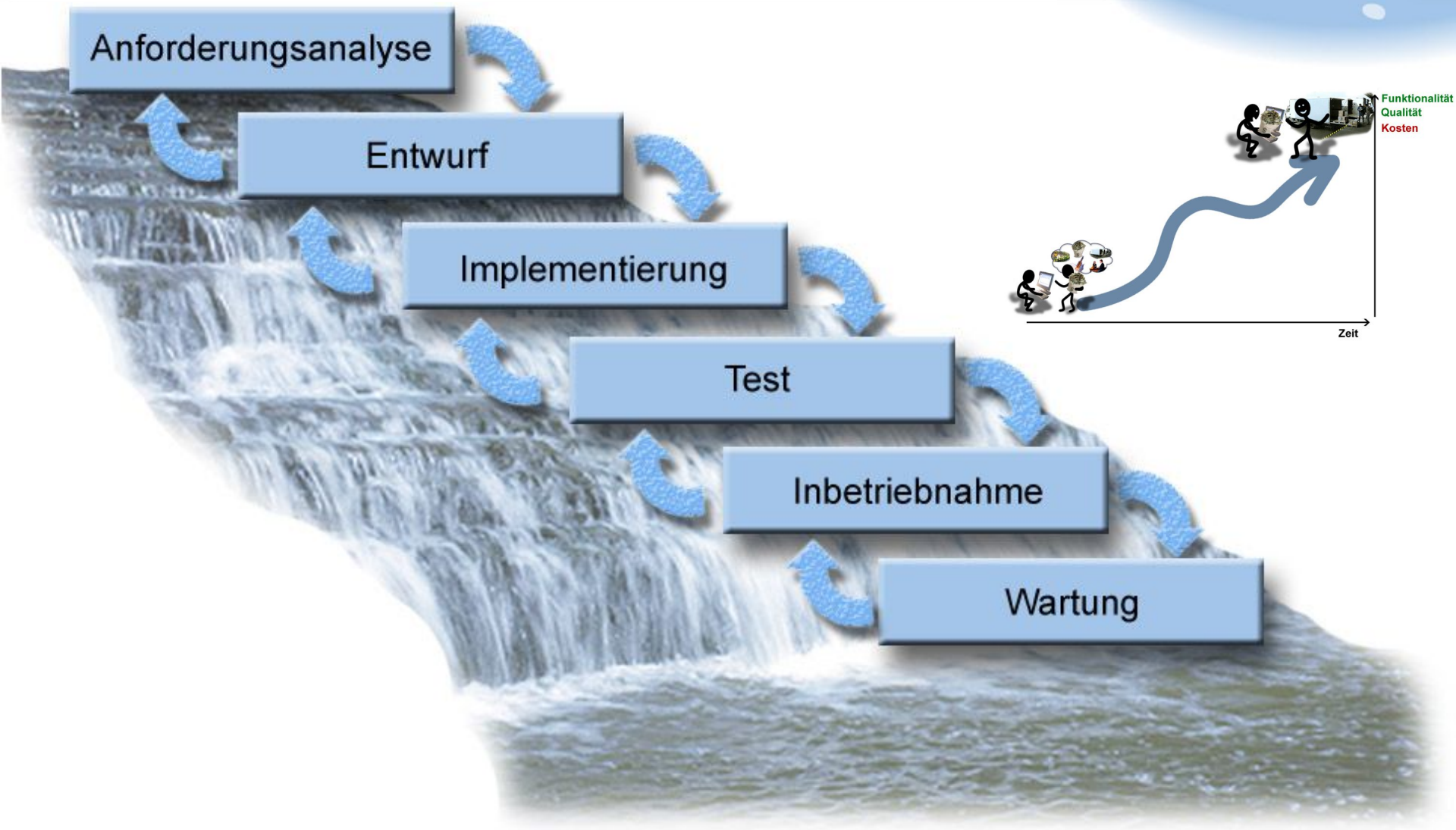
- Definierte Phasen
- Neben Implementierung, werden auch Analyse, Testen etc. explizit berücksichtigt.
- Gute Grundlage für Projektplanung

Schlecht:

- sehr idealisiert, reale Welt braucht oft mehrere Schleifen
- Anforderungen ändern sich
- neue Erfahrungen/Erkenntnisse schwer zu berücksichtigen
- „Planwirtschaft“

Wasserfallmodell mit Rücksprung

Definition



Phasen im Software-Engineering

Definition

Im Software-Engineering unterscheidet man verschiedene **Phasen**, wie z.B. **Planung, Anforderungsanalyse, Entwurf, Implementierung, Integration, Inbetriebnahme und Wartung.** Bzgl. der Anzahl und genauen Bezeichnung der Phasen gibt es viele Varianten.

Abhängig vom **Vorgehensmodell** werden die Phasen verschieden abgearbeitet. Unterschiede gibt es z.B. bzgl. der Bedeutung der einzelnen Phasen, ihrer zeitlichen Reihenfolge und Abhängigkeiten, eventuelle Unterteilung in Teilabschnitte, Zyklen usw.

Phasen im Software-Engineering

Übung

Übung

3.2 Phasen im Software-Engineering

Im Software-Engineering unterscheidet man typischerweise die folgenden Phasen: Planung, Anforderungsanalyse, Entwurf, Implementierung, Test, Inbetriebnahme und Wartung.

a) Überlegen Sie, welche Aktivitäten in den einzelnen Phasen stattfinden könnten.

b) Überlegen Sie, welche Personen/Rollen (z.B. Kunde, Software-Entwickler, Architekt, Business Analyst, Unterlieferant …), in den einzelnen Phasen mitarbeiten sollten und welche Aufgabe sie dort haben.

c) Was ist wichtig für die einzelnen Phasen? (Was tun? Was lassen?)

d) Was passiert, wenn eine Phase weggelassen/nicht richtig bearbeitet wird? Gehen Sie dazu der Reihe nach die Phasen durch.

Planung und Machbarkeitsanalyse

Definition

Bevor in einem Projekt in eine detaillierte Anforderungsanalyse durchgeführt wird, sollte eine grobe **Planung** aller Projektphasen erfolgen bzw. die grundsätzliche **Machbarkeit** des Projektes untersucht werden.

- Bei kleinen Projekten kann die erste Planung u.U. sehr schnell erfolgen
- Bei großen Projekten sind möglicherweise umfangreiche Vorstudien, Technologieuntersuchungen, Wirtschaftlichkeitsanalysen usw. erforderlich, die selbst wiederum als kleines Projekt gemanaget werden sollten.
- **Ergebnisse:** Wirtschaftlichkeitsbetrachtung, Go/No-go-Entscheidung, ein erster, grober Projektplan, Lastenheft (seitens des Kunden).

Anforderungsanalyse

Definition

Die **Anforderungsanalyse (requirements engineering)** hat zum Ziel, alle wesentlichen Anforderungen an das zu entwickelnde System systematisch zu ermitteln und zu dokumentieren.

- Neben den Anforderungen des Autraggebers sind in der Regel auch unternehmensinterne Anforderungen (z.B. verwendete Standardtechnologien, Unternehmens-/Produktstrategie) zu berücksichtigen.
- Anforderungen werden typischerweise in Prioritäten eingestuft (z.B. 1="Muss", 2="wünschenswert", 3="optional")
- **Ergebnis:** Pflichtenheft, welches als Bestandteil des Vertrages mit dem Kunden verbindlich ist.

Entwurf

Definition

Der **Entwurf (engl. design)** dient dazu, auf Basis der Anforderungen schrittweise eine entsprechende technische Realisierung zu entwerfen. Hierzu wird das System schrittweise vom Groben ins Feine in Komponenten und Verantwortlichkeiten zerlegt.

- Der Entwurf erfolgt auf verschiedenen Ebenen (je komplexer das System, desto mehr Ebenen).
- Typische Begriffe sind Systementwurf (=Gesamtsystem), Architektur (Gesamtsystem oder Teilsysteme / Komponenten), Design (auf Komponenten- / Modulebene). (Die Grenzen der Begriffe sind fließend.)
- Der Fokus liegt auf der Aufteilung in Komponenten/Module, Verantwortlichkeiten und Schnittstellen untereinander – nicht auf deren internen Realisierung.
- **Ergebnisse:** Entwurfsdokumente, Diagramme (z.B. UML)

Ende des Entwurfs

Übung

Übung

3.3 Ende des Entwurfs

Mit den Phasen Planung, Anforderungsanalyse und Entwurf sind prinzipiell alle Vorarbeiten erfolgt, um die eigentliche Programmierung zu beginnen.

a) Überlegen Sie, wieviel Zeit/Arbeitsaufwand des Projektes in diese Phasen investiert werden sollte.

b) Die strikte Trennung aller Phasen ist in der Praxis selten. Nennen Sie Gründe, warum die Grenzen zwischen den frühen Projektphasen bzw. zwischen dem Entwurf und der Implementierung verwischen könnten.

c) Wann wäre Ihrer Meinung nach ein Schritt zurück von einer späteren zu einer früheren Phase sinnvoll, wann nicht? Nennen Sie Beispiele.

Implementierung

Definition

Die **Implementierung (implementation)** ist die Umsetzung des Entwurfs durch Programmierung (Codierung) der einzelnen Komponenten / Module.

- Zur Implementierung gehört auch die Durchführung von grundlegenden Tests auf Modulebene sowie die Dokumentation des Quellcodes.
- Während der Implementierung können noch Entwurfsentscheidungen innerhalb der Module erforderlich sein. Die Schnittstellen / Verträge zwischen den Komponenten sollten aber möglichst nicht mehr verändert werden müssen.
- **Ergebnisse:** Dokumentierter Quellcode, kompilierte Komponenten/Systemteile, Testprotokolle.

Integration

Definition

Während der **Integration** werden die einzelnen Komponenten zum ablauffähigen Gesamtsytem zusammengesetzt und getestet.

- Das Kompilieren alle Quelldateien und anschließende Verbinden (Linken), Einbinden von Libraries usw. zum Gesamtsystem bezeichnet man als Build-Prozess. Bei großen Systemen kann der Buildprozess sehr komplex und aufwändig werden, so dass es z.B. spezielle Mitarbeiter gibt, die ausschließlich für den Buildprozess verantwortlich sind.
- Nach dem Build erfolgt überlicherweise die Installation der Software auf einem Testsystem für die Durchführung des Integrationstests.
- **Ergebnisse:** Ablauffähiges Gesamtsystem, Protokolle vom Build und den Integrationstests

Übung

Übung

3.4 Implementierung und Integration

a) Die Phasen Implementierung und Integration sind in der Praxis typischerweise eng verzahnt. Warum?

b) Was könnten wichtige Qualifikationen für Mitarbeiter sein, die in der Implementierung bzw. in der Integration arbeiten? Worin unterscheiden sich die Anforderungsprofile?

c) Manche Softwarefirmen praktizieren „Daily Builds“. Was versteht man darunter? Welche Vorteile und Nachteile könnte es haben?

Inbetriebnahme

Definition

Bei der **Installation und Inbetriebnahme (Deployment)** wird die Software auf dem Zielsystem installiert, konfiguriert und ggf. abschließend getestet. Nach erfolgreicher Installation kann der produktive Einsatz der Software beginnen.

- Grundsätzlich sind zwei Szenarien zu unterscheiden:
  - Standardsoftware, die für einen breiten Markt bestimmt ist
  - Kundenspezifische Software, die eine maßgeschneiderte Lösung für das Kundenproblem darstellt.
- Vor der endgültigen Auslieferung (Release) einer Software werden oft Vorabversionen zu Testzwecken ausgeliefert:
  - Stichwörter: Alpha, Beta, Release Candidate, Release

Inbetriebnahme

Übung

Übung

3.5 Inbetriebnahme

Die Inbetriebnahme von maßgeschneiderter Software für einen speziellen Kunden und von Standardsoftware für einen breiten Markt weist wichtige Unterschiede auf. Untersuchen Sie diese Unterschiede:

a) Nennen Sie Beispiele für beide Fälle.

b) Diskutieren Sie die unterschiedlichen Anforderungen und mögliche Probleme in beiden Fällen. Was ist im Vergleich zwischen den beiden Szenarien jeweils einfacher oder schwieriger?

Wartung

Definition

Nach der ersten Installation der Software und bis ihrem Einsatzende fallen regelmäßig Aufgaben wie Fehlerkorrekturen, Anpassungen oder kleinere Erweiterungen an. Diese werden als **Wartung** bezeichnet.

- Wartung macht oft einen Großteil der Gesamtkosten für die Softwareentwicklung aus.
- Heutzutage ist wohl keine frisch releaste Software „so fertig“, dass sie danach komplett ohne Wartung auskäme.
- Beispiele für Betriebssystemwartung: Microsoft-Patches und Updates.
- Bei größerem Änderungsbedarf, z.B. Entwicklung einer komplett neuen Produktversion spricht man nicht mehr von Wartung.
- **Ergebnis:** Ein System, das stets auf dem neuesten Stand ist und mit den aktuellen Technologien, Hardware, Betriebssystemversionen usw. funktioniert.

Übung

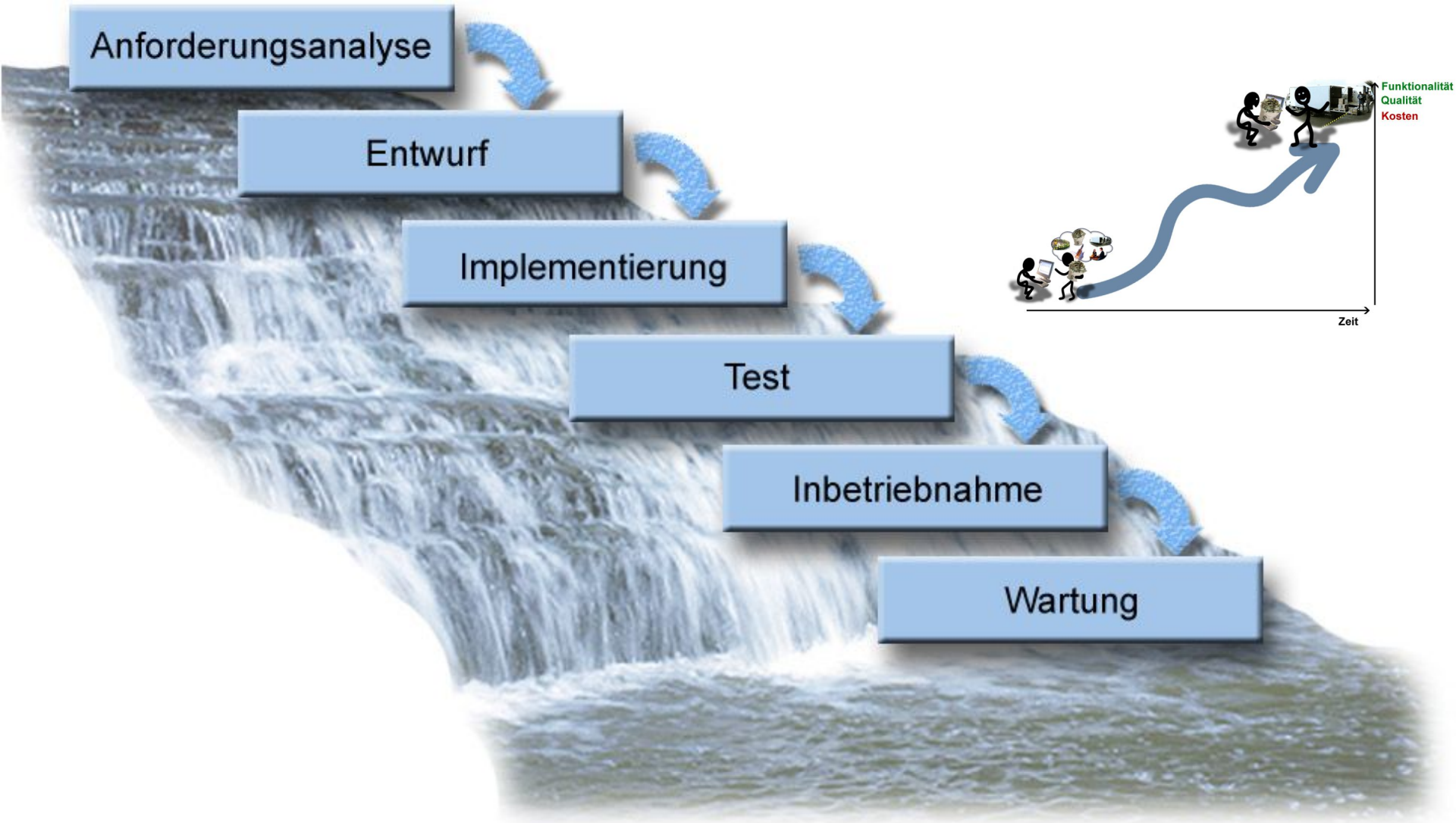
Übung

3.6 Wartung

a) Recherchen Sie im Internet über Software-Wartung (Software maintenance). Welche Aussagen finden Sie dort, z.B. über die Bedeutung der Wartung, Kosten und Verfahren.

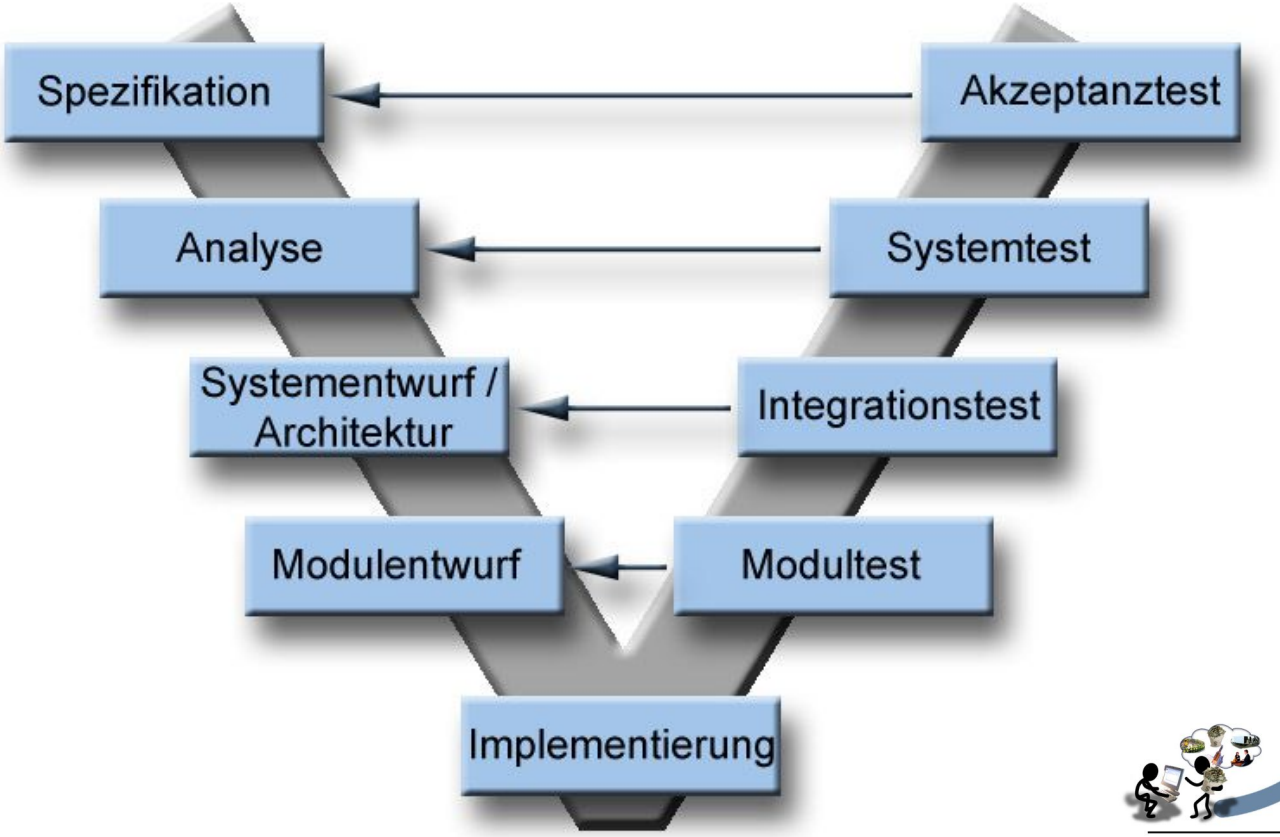
b) Von welchen Faktoren hängen Wartungskosten ab? Wie kann man Wartungskosten senken?

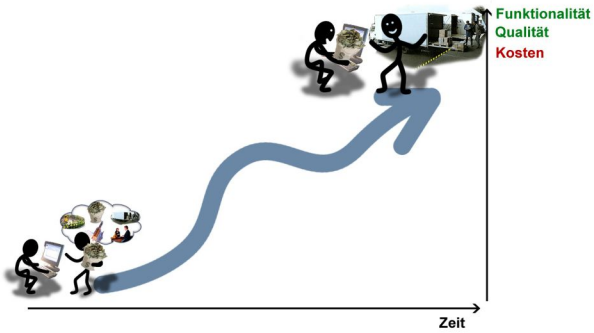
Wasserfallmodell



V-Modell

Definition





V-Modell

Übung

Übung

3.7 V-Modell

Untersuchen Sie das V-Modell:

a) Recherchieren Sie im Internet nach dem V-Modell. Was für Informationen und Meinungen finden Sie? Gibt es Varianten? Ist das V-Modell „modern“?

b) Wie würden Sie das Modell beurteilen? Diskutieren Sie Vor- und Nachteile des Modells.

V-Modell XT

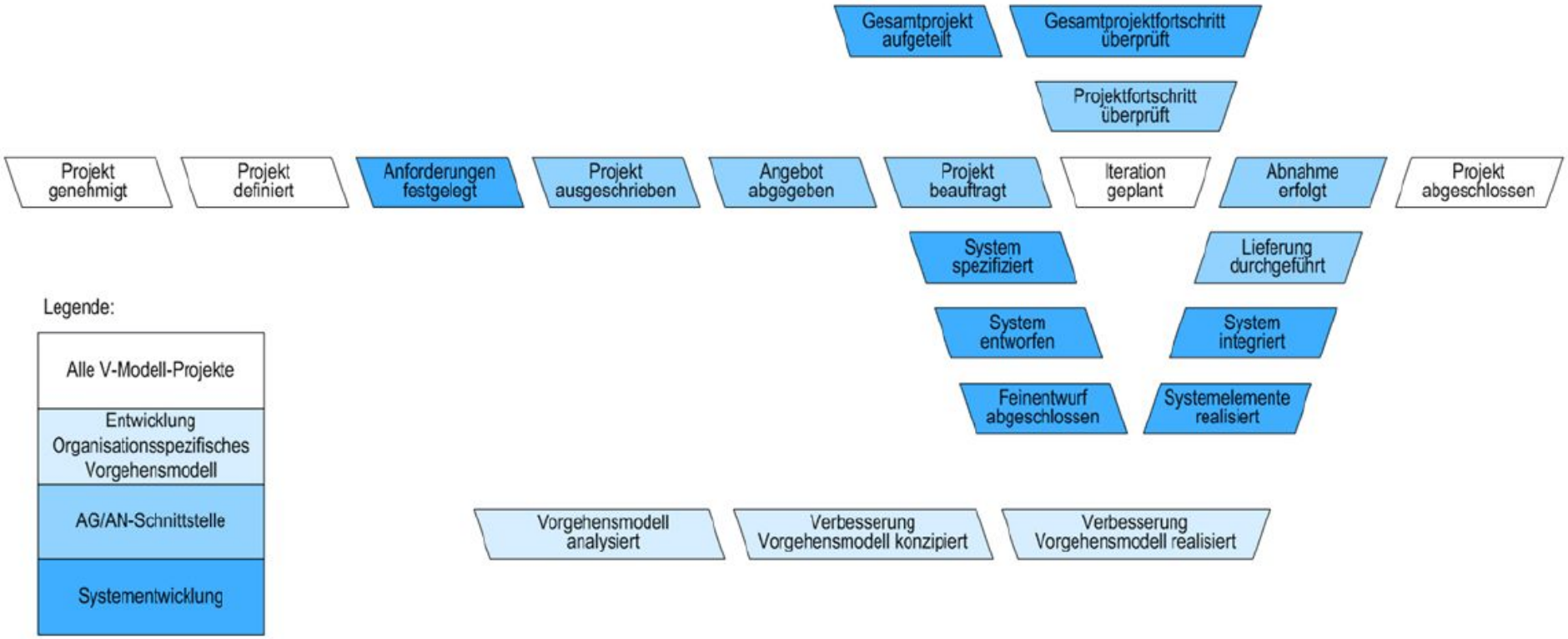
Übung

- Ursprung: Modell für SW-Projekte bei den Bundesbehörden. (parallele unabhängige Entwicklung ähnlicher Konzepte aber auch in den USA)

Aktueller Standard: V-Modell XT

- Zentrale Aspekte: Rollen, Produkte und Aktivitäten.
- XT = Xtreme Tailoring: Modulare, flexible Vorgehensbausteine zur Anpassung an Projekttypen
- Mehrere Projektdurchführungsstrategien, z.B. inkrementelle, agile und komponentenorientierte Entwicklung
  - (Teil-)Produkte müssen zu Entscheidungspunkten vorliegen (genaue Entwicklungsreihenfolge wird nicht betrachtet)
- Zusätzliche Themen: Migration, Ergonomie, Systemsicherheit, Ausschreibung, Vertragsschluss, Abnahme

V-Modell XT


Gesamtprojekt aufgeteilt
Gesamtprojektfortschritt überprüft
Projektfortschritt überprüft
Projekt genehmigt
Projekt definiert
Anforderungen festgelegt
Projekt ausgeschrieben
Angebot abgegeben
Projekt beauftragt
Iteration geplant
Abnahme erfolgt
Projekt abgeschlossen
System spezifiziert
Lieferung durchgeführt
System entworfen
System integriert
Feinentwurf abgeschlossen
Systemelemente realisiert
Legende:
Alle V-Modell-Projekte
Entwicklung Organisationsspezifisches Vorgehensmodell
AG/AN-Schnittstelle
Systementwicklung
Vorgehensmodell analysiert
Verbesserung Vorgehensmodell konzipiert
Verbesserung Vorgehensmodell realisiert

V-Modell XT - Rollen

Beispiel

- Akquisiteur
- Änderungssteuerungsgruppe (Change Control Board)
- Änderungsverantwortlicher
- Anforderungsanalytiker (AG)
- Anforderungsanalytiker (AN)
- Anwender
- Assessor
- Ausschreibungsverantwortlicher
- Einkäufer
- Ergonomieverantwortlicher
- HW-Architekt
- HW-Entwickler
- KM-Administrator
- KM-Verantwortlicher
- Lenkungsausschuss
- Logistikentwickler
- Logistikverantwortlicher
- Projektkaufmann
- Projektleiter
- Projektmanager
- Prozessingenieur
- Prüfer
- QS-Verantwortlicher
- Qualitätsmanager
- SW-Architekt
- SW-Entwickler
- Systemarchitekt
- Systemintegrator
- Systemsicherheitsbeauftragter
- Technischer Autor

Rolle: Anforderungsanalytiker

Beispiel

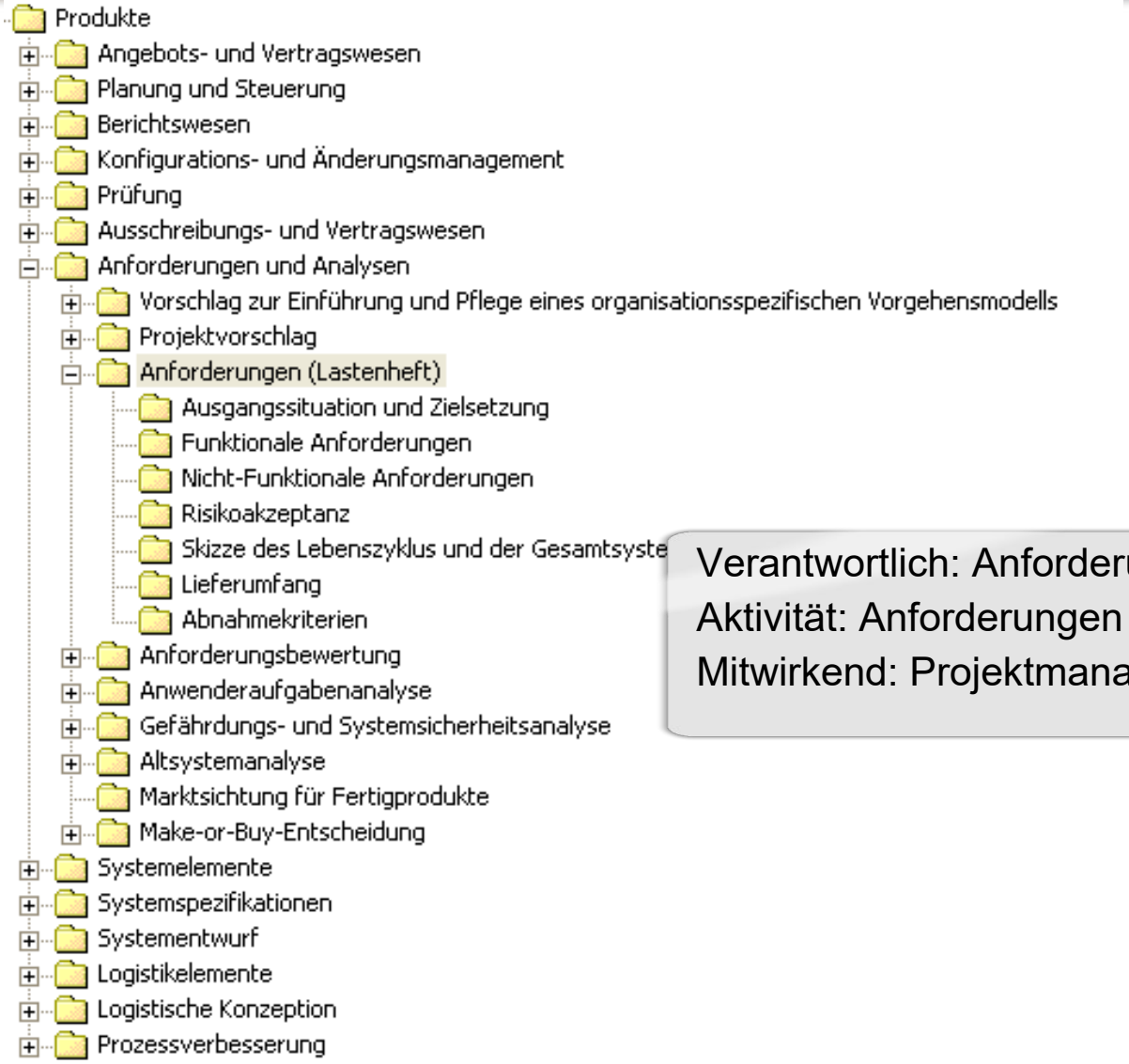
Der Anforderungsanalytiker (AG) ist nach erfolgreicher Beauftragung des Projekts für die Erstellung des Produktes Anforderungen (Lastenheft) verantwortlich. Er hat die Qualität der Anwenderanforderungen sicherzustellen und die Voraussetzungen für die Verfolgbarkeit und die Veränderbarkeit der Anforderungen über alle Lebenszyklusabschnitte zu schaffen. Der Anforderungsanalytiker (AG) hat die Grundlagen des Fachgebietes "Requirements Engineering" bei der Aufgabendurchführung zu beachten.

Aufgaben und Befugnisse

- Erarbeiten der Grundlagen für die Erstellung und das Management von Anforderungen,
- Auswahl und Einrichten der Werkzeuge für die Erfassung und Verwaltung der Anforderungen,
- Analyse von Geschäftsprozessen,
- Mitarbeit bei Realisierungsuntersuchungen,
- Analyse von Bedrohung und Risiko,
- Durchführung von Schwachstellenanalyse und Sicherheits- und Leistungsanalyse,
- Erfassen und Beschreiben der funktionalen und nicht-funktionalen Anforderungen,
- Abstimmen und Harmonisieren der erfassten Anforderungen mit allen Beteiligten,
- Systematisieren und Priorisieren der erfassten Anforderungen,
- Erstellen von Abnahmekriterien,
- **Erstellen des Entwurfs eines Anforderungsdokuments,**
- Qualitätssicherndes Überprüfen der Anforderungen nach vorgegebenen Qualitätskriterien,
- Überprüfen des Systementwurfs auf Einhaltung der Anwenderanforderungen,
- Mängelbeseitigung bei Anforderungen,
- Aufbereiten der Anforderungen für das Anforderungscontrolling,
- Bewerten von Anforderungen nach vorgegebenen Kriterien,
- Analyse der operationellen Notwendigkeit und der technischen Machbarkeit von Anforderungen,
- Bewerten der Anforderungen nach deren Wirtschaftlichkeit (Kosten-Nutzen-Analysen),
- Erstellen eines ausschreibungsreifen Anforderungsdokumentes.

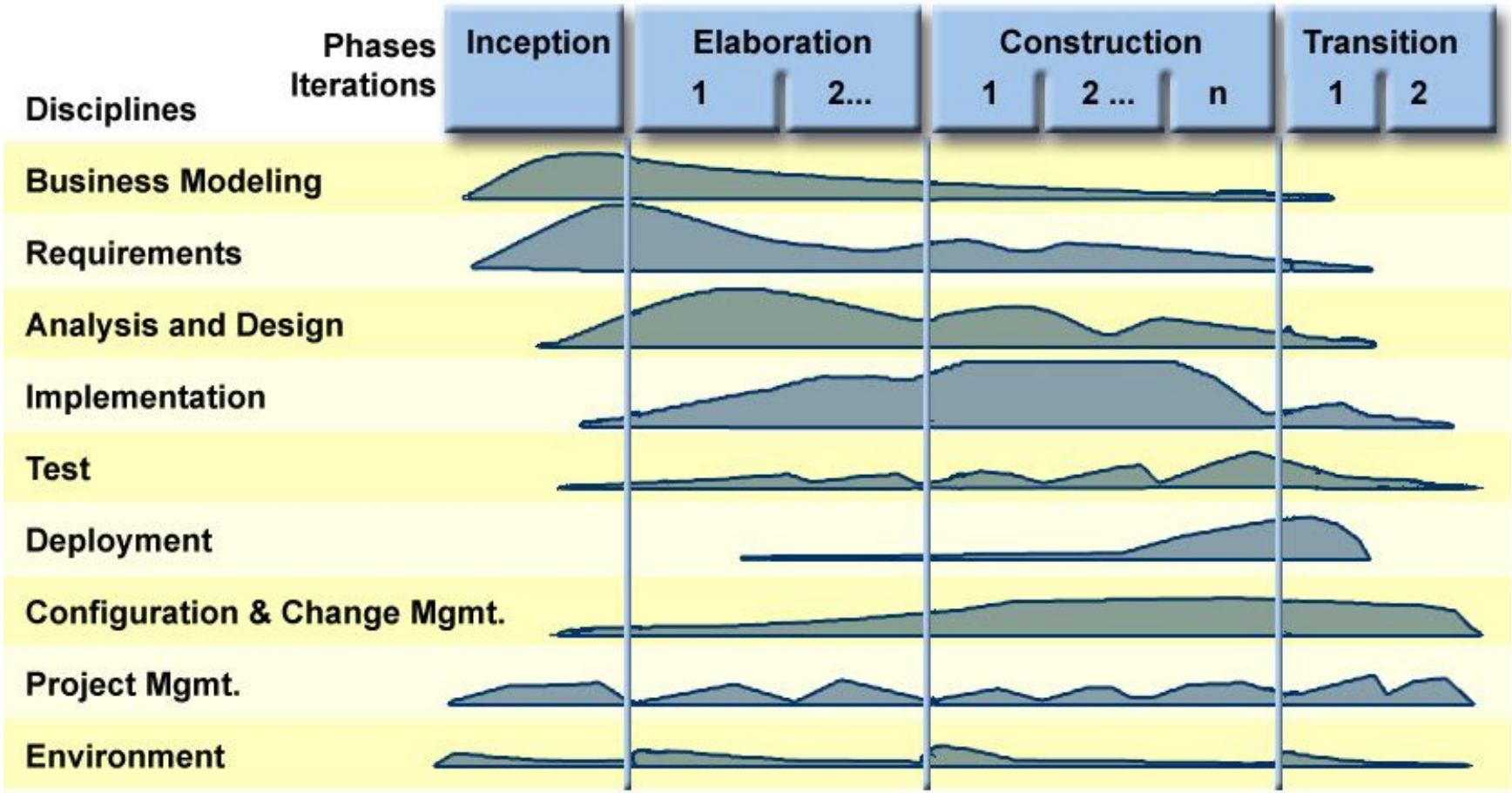
V-Modell XT - Produkte

Beispiel



Verantwortlich: Anforderungsanalytiker (Auftraggeber)
Aktivität: Anforderungen festlegen
Mitwirkend: Projektmanager, Projektleiter, Anwender

RUP Entwicklungsmodell*



*** RUP = Rational Unified Process**

RUP - Erläuterungen

- Business Modeling: Beschreibung des geschäftlichen Umfeldes, Geschäftsprozesse, Abläufe
- Requirements: Anforderungsanalyse
- Analysis/Design: Semiformale/grafische Beschreibung der Anforderungen (z.B. UML Use Cases), Grobentwurf Systemarchitektur usw. (verschiedene Sichten)
- Implementation/Test
- Deployment: s. Inbetriebnahme
- Configuration & Change Management: Verwaltung der verschiedenen Versionen des Softwarequellcodes oder anderer Komponenten.
- Project Management: Steuerung des Entwicklungsprozesses
- Environment: Bereitstellen der Entwicklungsumgebung und benötigter Ressourcen (Rechner, Werkzeuge, …)

Übung

Übung

3.8 Rational Unified Process

Untersuchen Sie das RUP-Vorgehensmodell:

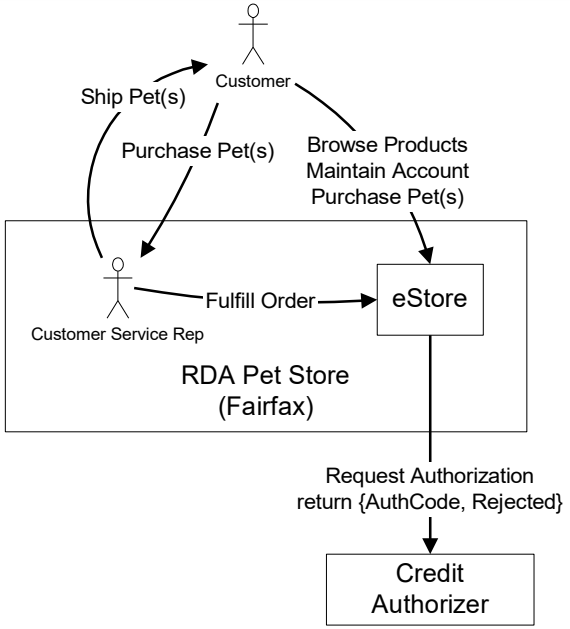
a) Recherchieren Sie im Internet nach RUP. Was für Informationen und Meinungen finden Sie?

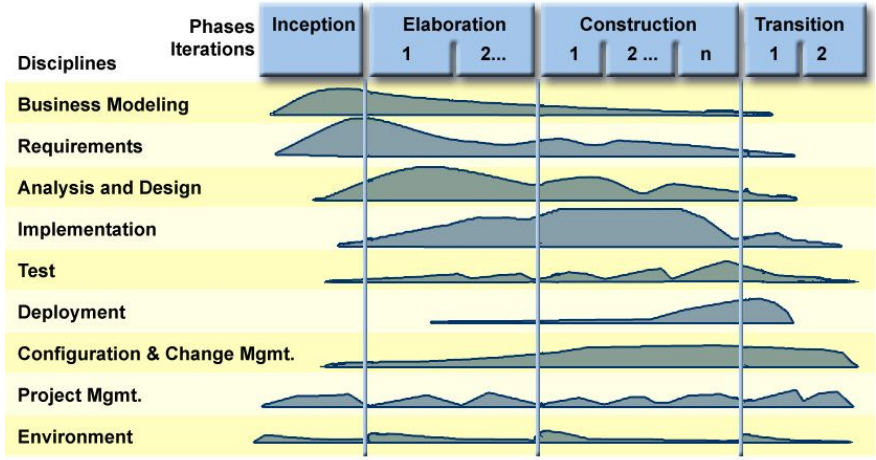
b) Nennen Sie Unterscheide zu den anderen Modellen. Welche zusätzlichen Aspekte gibt es? Was erscheint Ihnen wichtig?

c) Nennen Sie Vor- und Nachteile des Modells im Vergleich mit den anderen vorgestellten Modellen.

RUP

- Fokus auf Anwendungsfälle (Use Cases) und Architektur
- Objektorientierung und Beschreibung mit UML
- Iteratives, inkrementelles Vorgehen
- Rollen (roles), Produkte (products), Aufgaben (tasks)
- RUP ist ein "Framework", in dessen Rahmen verschiedene Vorgehensweisen eingebettet werden können.
- von Rational (nun bei IBM) entwickelt: IBM bietet Werkzeuge zur Unterstützung an.
- Es gibt Varianten/Ableger, z.B. Open Unified Process





Bekannte (Prozess-)Probleme?

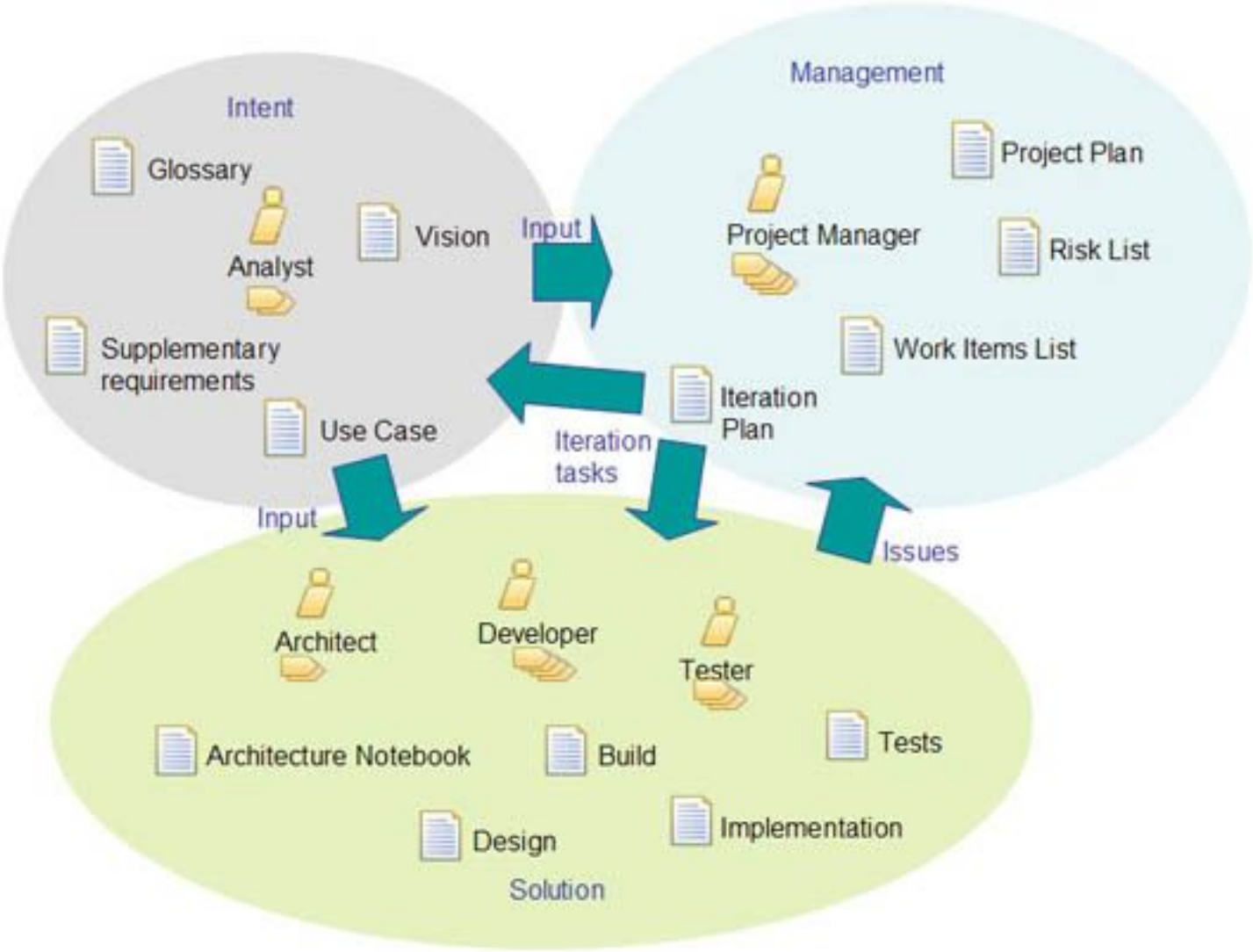
Auszug aus **OpenUP –The Best of Two Worlds, Bjorn Gustafsson**
http://www.methodsandtools.com/archive/archive.php?id=69

Some common problems can be observed in troubled projects:

- **The project team doesn't share a clear vision of how the system will appear to its users.** Without a clear vision of the final system, there is no guiding framework for the work in the project. The team's analysts have no means to calibrate their requirements to the scope and effort of the project, which results in ill-fitting requirements statements; and the development team can not properly prioritize their work.
- **Requirements do not drive development work.** Some development cultures regard requirements as "incidental" input to the project only, and drive the development work based on other, internal and technical, conditions. This is commonly found in "silo" organizations where there are separate teams for requirements capture and development.
- **The system's architecture has not been articulated.** Although projects that only evolve and maintain existing systems may not need to pay much attention to architecture, there are many projects that do. As Dean Leffingwell points out: "… how much architecture a team needs depends in large part on what the team is building".
- **Plans are not connected to the engineering reality.** Plans are often created and maintained separately from the actual project work. We have all seen nebulous Gantt charts that project managers spent days or even weeks creating, with hundreds of line items in nifty breakdown structures, purportedly believed to bring the project to "completion" at some well-defined point in time. Of course, these plans become outdated even before they are pinned on the wall.
- **Risks are ignored.** All projects run the risk of building the wrong product or not being able to build the product as envisioned, yet very few projects acknowledge this uncertainty and actively try to reduce it.

OpenUP Elements

Beispiel



aus: http://www.methodsandtools.com/archive/archive.php?id=69p3

Agile Methoden

Definition

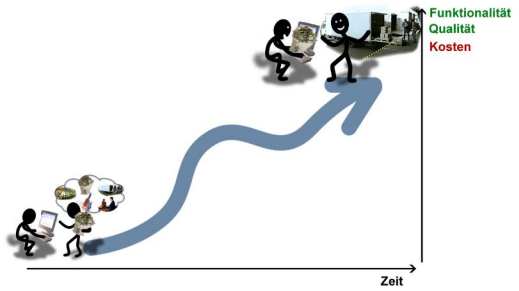
Agile Methoden (agile methods) sind ein Sammelbegriff für verschiedene Vorgehensmodelle, die die folgenden Punkte betonen:

- Teamarbeit (überschaubare Teams), enge Zusammenarbeit auch mit dem Kunden
- Kommunikation wichtiger als Dokumentation
- Inkrementelle Entwicklung: Viele kleine Entwicklungsschritte mit wenig Planung
- Iteration (Wiederholung): In jedem Schritt wird von Anforderungsanalyse bis Testen alles im "Miniaturformat" durchlaufen. Am Ende steht immer ein ausführbares Programm (wenn auch anfangs mit minimaler Funktionalität)
- Leichtgewichtige Prozessmodelle, minimale langfristige Planung, Flexibilität.

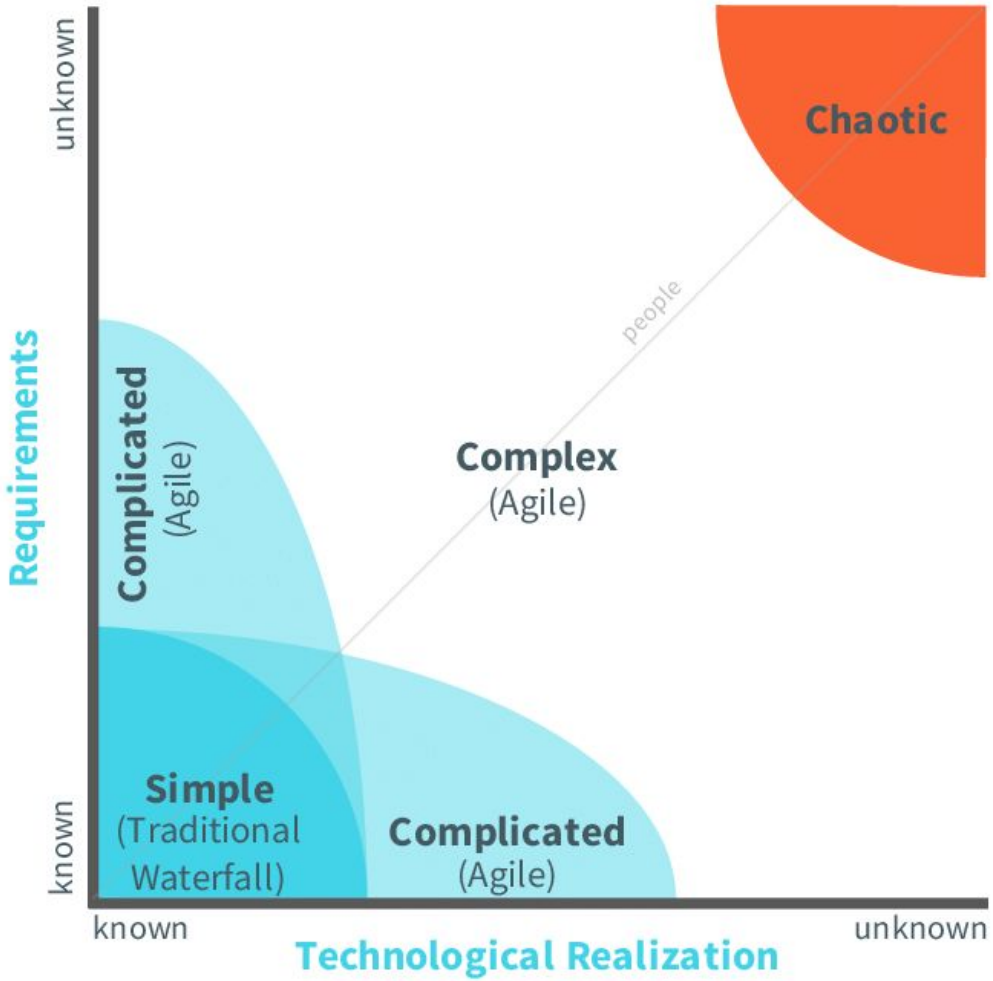
Beispiele:

- Scrum
- XP - Extreme Programming
- FDD – Feature Driven Development
- RAD - Rapid application development
- Crystal Clear Software Development
- DSDM - Dynamic Systems Development Method

→ Agile Methoden waren/sind auch eine Reaktion auf übertrieben bürokratische, ineffiziente Entwicklungsprozesse mit Übermaß von Dokumentation, die niemand liest.



Stacey Matrix für Software-Entwicklung



Bildquelle: Ömer Uludag,TU München

Agile Methoden

Übung

Übung

3.9 Agile Methoden

Recherchen Sie im Internet nach agilen Vorgehensmodellen.

a) Welche Methoden finden Sie (Namen der Methoden)? Welche scheinen verbreitet, welche eher selten?

b) Worin unterscheiden sich die Ansätze?

c) Welche Eigenschaften/Mechanismen werden als positiv herausgestellt? Was wird kritisch diskutiert?

Agile Manifesto

2001: Treffen von Schlüsselpersonen der agilen Softwareentwicklung:
http://agilemanifesto.org/

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

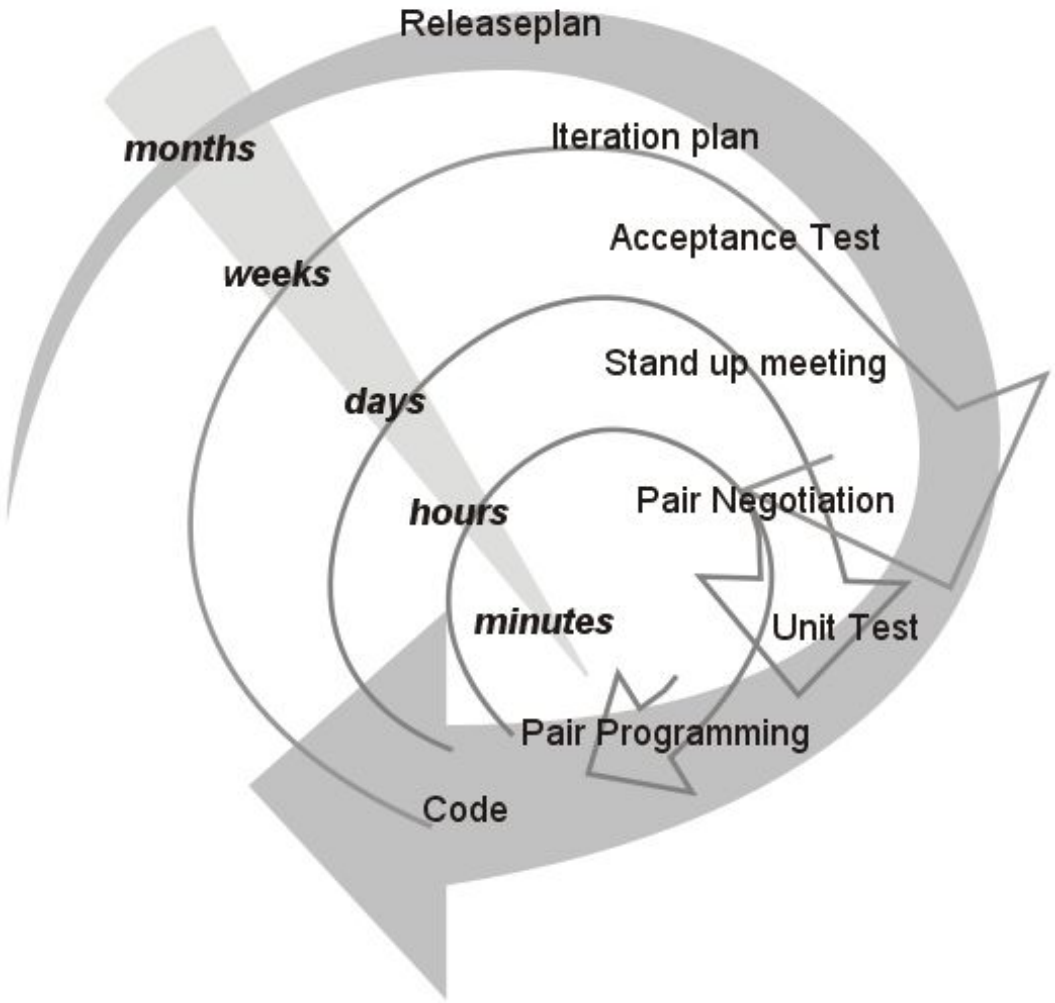
Agile Manifesto - Prinzipien

2001: Treffen von Schlüsselpersonen der agilen Softwareentwicklung:
http://agilemanifesto.org/

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

XP - Extreme Programming

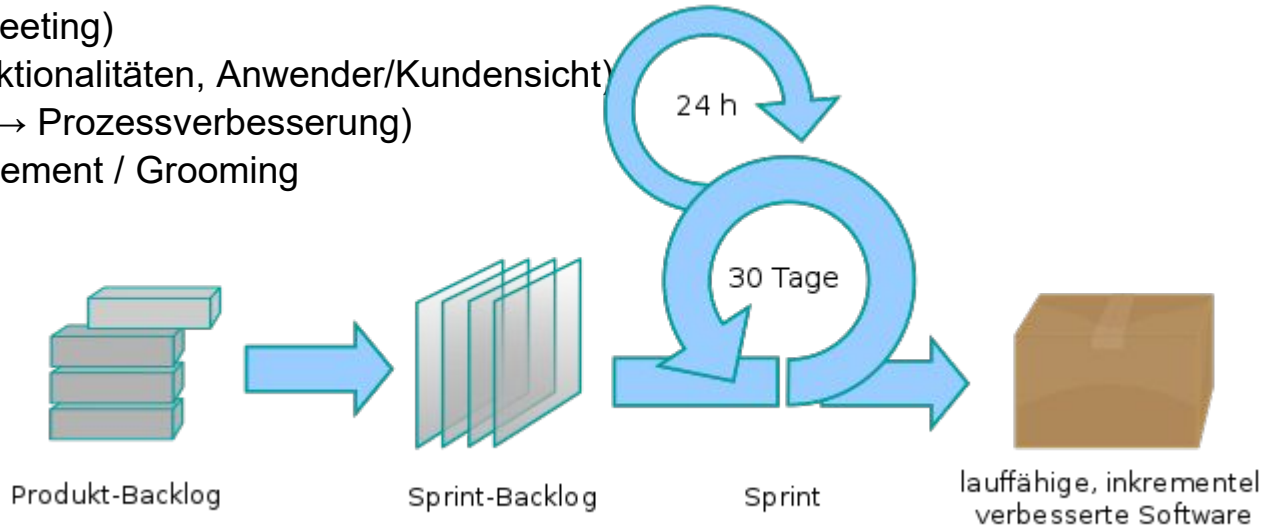
Beispiel



http://en.wikipedia.org/wiki/Extreme_Programming

Scrum

- Agiles Vorgehensmodell, weit verbreitet
- Inkrementell: „Sprints“ (30 Tage)
- Artefakte:
  - Product Backlog (Anforderungsliste)
  - Sprint Backlog (Plan für Sprint)
  - Product Increment (Sprintergebnisse)
- Aktivitäten:
  - Sprint Planning (was+wie)
  - Daily Scrum (15min-Meeting)
  - Sprint Review (→ Funktionalitäten, Anwender/Kundensicht)
  - Sprint Retrospective (→ Prozessverbesserung)
  - Product Backlog Refinement / Grooming
- Rollen
  - Product Owner (Anforderungen, Kundensicht)
  - Entwicklungsteam
  - Scrum Master („Prozessmoderator“)
- Teamgröße<10, Werkverträge...?



Bildquelle: https://de.wikipedia.org/wiki/Scrum#/media/File:Scrum_process-de.svg

Vorsorgen für später?

Übung

Übung

3.10 Vorsorgen für später?

Im agilen Vorgehen wird propagiert, dass man nur die tatsächlich angeforderten Funktionen entwirft und implementiert, anstatt zu versuchen, Mechanismen vorzusehen, die eventuelle zukünftige Erweiterungen erleichtern sollen.

a) Diskutieren Sie Vor- und Nachteile dieses Ansatzes.

b) Was ist Ihr Fazit, Ihre Empfehlung?

Vorsorgen für später?

Übung

- Programmierer versuchen nicht, bei der Realisierung des aktuellen Releases bereits künftige Releases mit zu berücksichtigen
- Lässt sich meist eh nicht genau vorhersehen
- Programmierer sind zum ständigen Umbau der bereits erstellten Software bereit
- Refactorings + Redesign
- Problem: Lokale Optimierung vs. optimale Systemarchitektur
- Rapid-Prototyping-Vorgehensweise

Wahl eines Vorgehensmodells

Entscheidungskriterien:

- Welches Modell wurde (im Unternehmen, vom Team) bisher verwendet? Wie erfahren ist das Team damit? Hat sich das Modell bewährt?
- Wie groß ist das Projekt? Wie hoch sind die Projektrisiken? Wie hoch sind die Qualitätsansprüche?
  - sicherheitskritische Lösungen erfordern z.B. zwingend einen starren, sehr genauen dokumentierten Prozess (→ Zertifizierung).
- Gibt es äußere Randbedingungen, unbedingt zu verwendende Standards?
  - Unternehmensrichtlinien, z.B. Unternehmen strebt CMMI-Zertifizierung an.
- Wie genau sind die Anforderungen am Anfang bekannt? Kann es noch größere Änderungen im Laufe des Projektes geben?
  - Extrembeispiel: Forschungsprojekte → Projektergebnis ist am Anfang vage
- Grundsatzentscheidung: Agil oder Traditionell?
- Was ist der richtige Umfang/Aufwand für den Prozess (welche Dokumente usw. sind erforderlich?)
- Sind Abweichungen vom Modell erlaubt? Vereinfachungen (z.B. bestimmte Phasen weglassen)?

Egal, welches Modell letzlich gewählt wird, wichtig ist:
Es gibt ein definiertes, geplantes Vorgehen und das Team versteht es und befolgt es!

Lieferant: Vorgehensmodell

Software-Engineering-Projekt

P.9 Lieferant: Vorgehensmodell

Sie haben den Auftrag erhalten und wollen nun entscheiden, welches Vorgehensmodell die Basis Ihrer Projektplanung werden soll.

a) Welches Vorgehensmodell erscheint Ihnen grundsätzlich geeignet? Warum? (Vorteile/Nachteile?)

b) Welche Teile des Vorgehensmodells wollen Sie verwenden, wo nehmen Sie Vereinfachungen oder Änderungen vor? Welche Bausteine des Modells sind für Sie wichtig, welche können Sie weglassen?

c) Der Qualitätsmanager Ihres Unternehmens, der auch für die Definition der internen Prozesse verantwortlich ist, möchte die Vorgehensweise in ihrem Projekt begutachten. Grundsätzlich hat er auch Bereitschaft signalisiert, für dieses Projekt von den Standardprozessen abzuweichen und etwas Neues auszuprobieren. Bereiten Sie ein kleine Präsentation (zwei, drei Folien) vor, um Ihren Qualitätsmanager von Ihrer Vorgehensweise zu überzeugen.