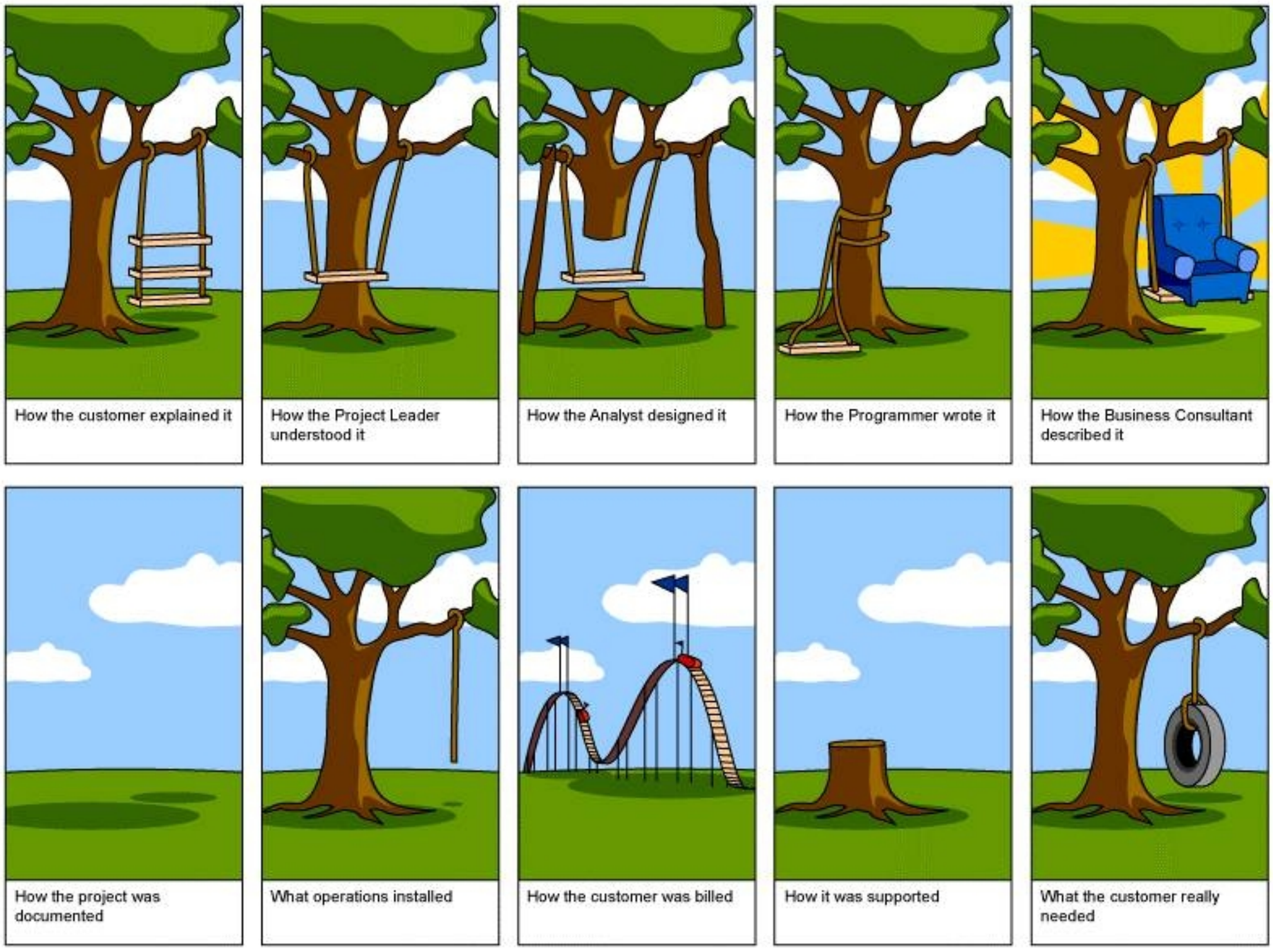
Softwaretechnik

Teil 1 - Planung

Motivation


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Was sind die Probleme?

Problemquellen bei Software-Projekten:

- Unvollständige Anforderungen 13.1%
- Kunden nicht ausreichend einbezogen 12.4 %
- Mittel nicht ausreichend 10.6 %
- Unrealistische Erwartungen 9.9 %
- Mangelnde Unterstützung durch Management 9.3 %
- Änderungen in den Anforderungen 8.7 %
- mangelnde Planung 8.1 %

nach [CHAOS Report, Standish Group 1995]

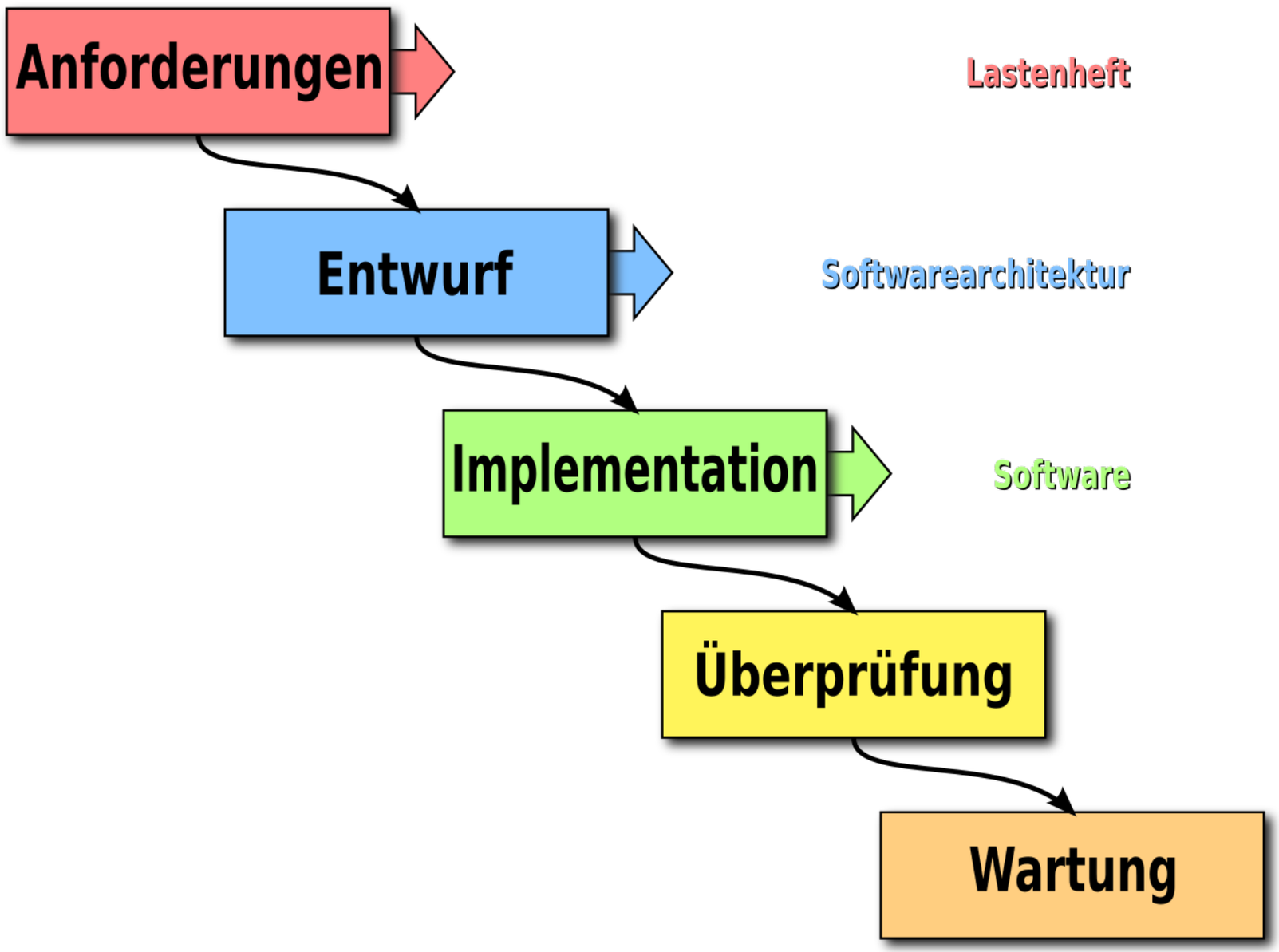
Projekte & Stakeholder

- Kunden verwenden eigene **Fachsprache**
- Kunden wissen oft nicht was sie wirklich wollen/brauchen
- Verschiedene **Stakeholder** mit **widersprüchlichen** Anforderungen
- Politische und organisatorische Faktoren können Anforderungen beeinflussen
- Anforderungen **ändern** sich während der Analyse und Entwicklung
- **Neue Stakeholder** mischen sich ein

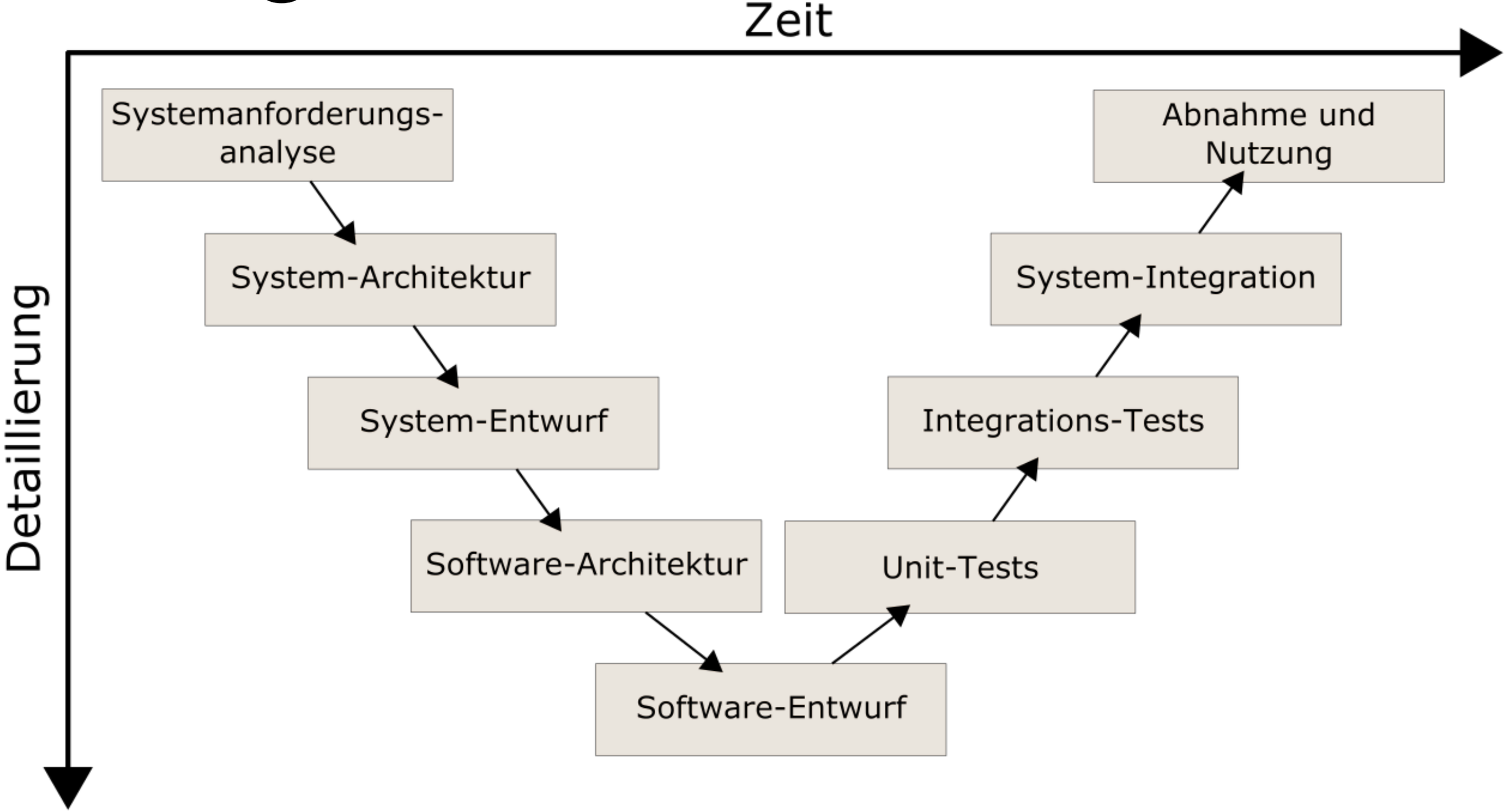
Stakeholder

- Einzelperson oder Organisation
- Sind am aktiv am Projekt **Interessierte/Beteiligte**
- **Beeinflussen** das Projekt und seine Ergebnisse
- Interessen werden als **Folge** des Projektverlaufs und -erfolges direkt **positiv** oder **negativ** beeinflusst

Klassische Vorgehensmodelle: Wasserfall



Klassische Vorgehensmodelle: V-Modell



Klassische Vorgehensmodelle: Spiralmodell

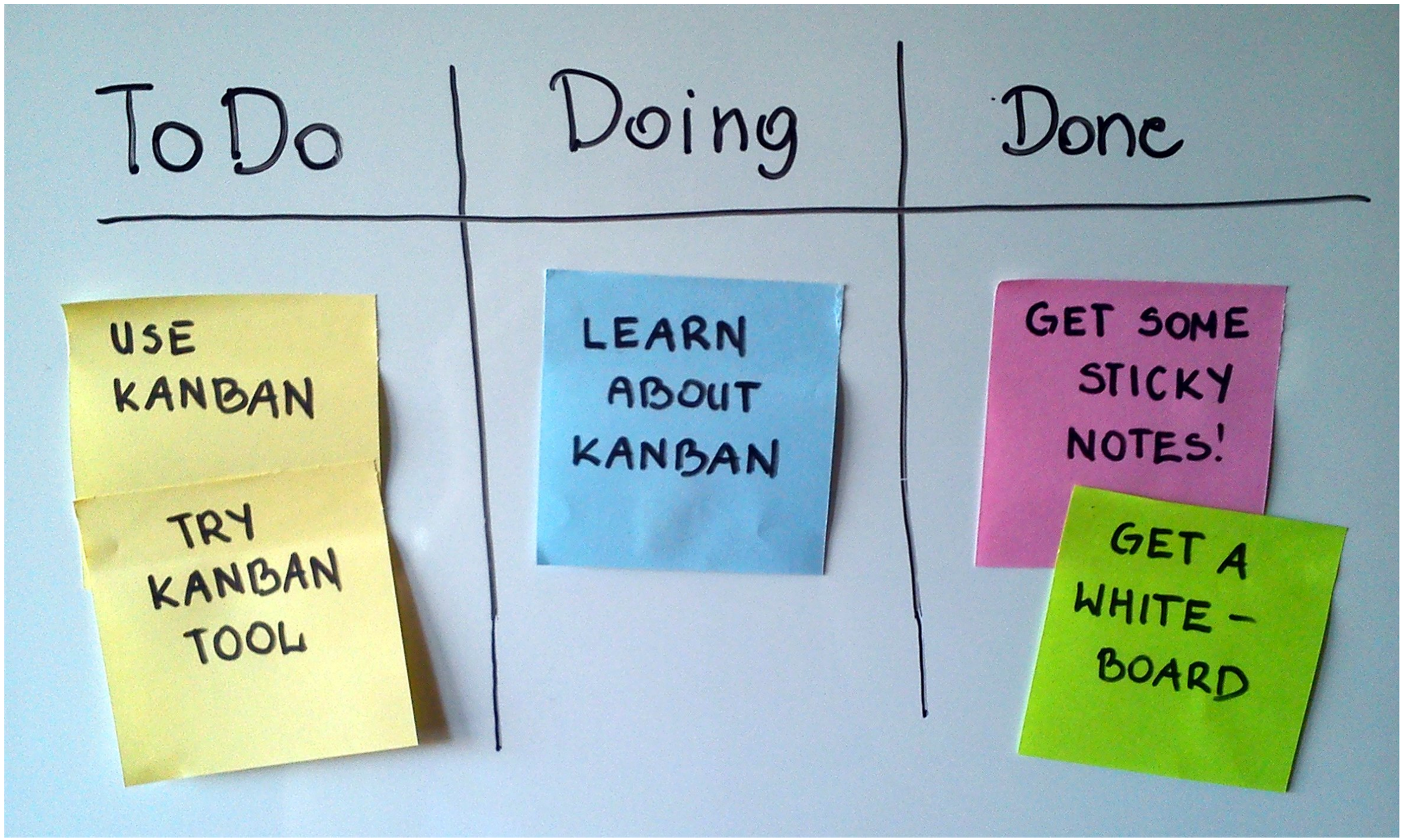
Vorgehensmodelle: Die “Agile” Bewegung

- **Individuen und Interaktionen** stehen über Prozessen und Werkzeugen
- **Funktionierende Software** steht über einer umfassenden Dokumentation
- **Zusammenarbeit mit dem Kunden** steht über der Vertragsverhandlung
- **Reagieren auf Veränderung** steht über dem Befolgen eines Plans

Kanban

- Visualisiere den Workflow, den Arbeitsfluss.
- Fang eine neue Aufgabe nicht an, bevor eine andere erledigt ist. Begrenze die Anzahl an Aufgaben in Bearbeitung!
- Messe die durchschnittliche Zeit zwischen Beginn und Fertigstellung einer Aufgabe. Finde Wege, diese Zeit zu verringern und möglichst konstant zu machen.

Kanban Tafel

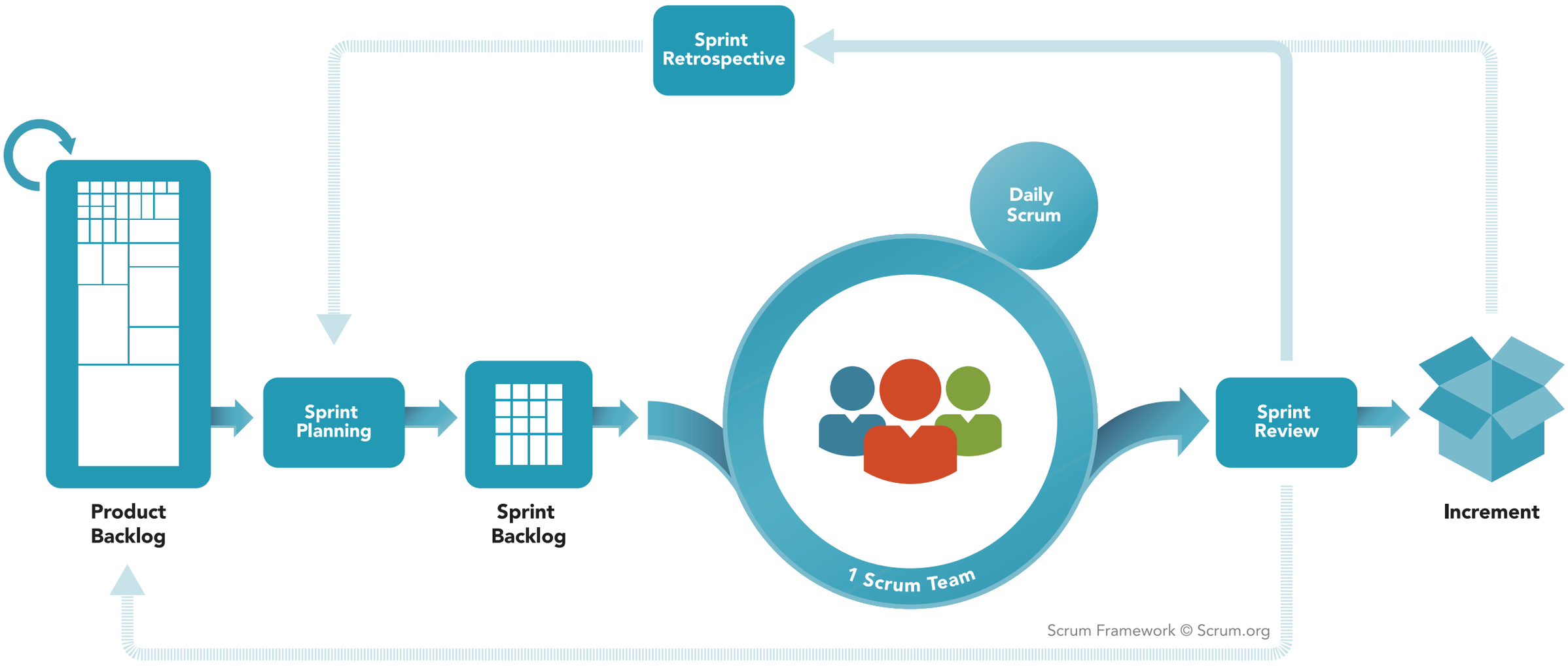


Agile Vorgehensmodelle: Scrum

- kleine, selbstorganisierende Teams
- Aufteilung des Projektergebnisses in Inkremente
- Aufteilung der Projektlaufzeit in kurze Iterationen (meist 1 bis 4 Wochen)
- Visualisierung der Aufgabenerledigung in einer Iteration
- Lernen und Anpassen von Projektziel und Arbeitsweise zwischen den Iterationen
- Guide: https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf

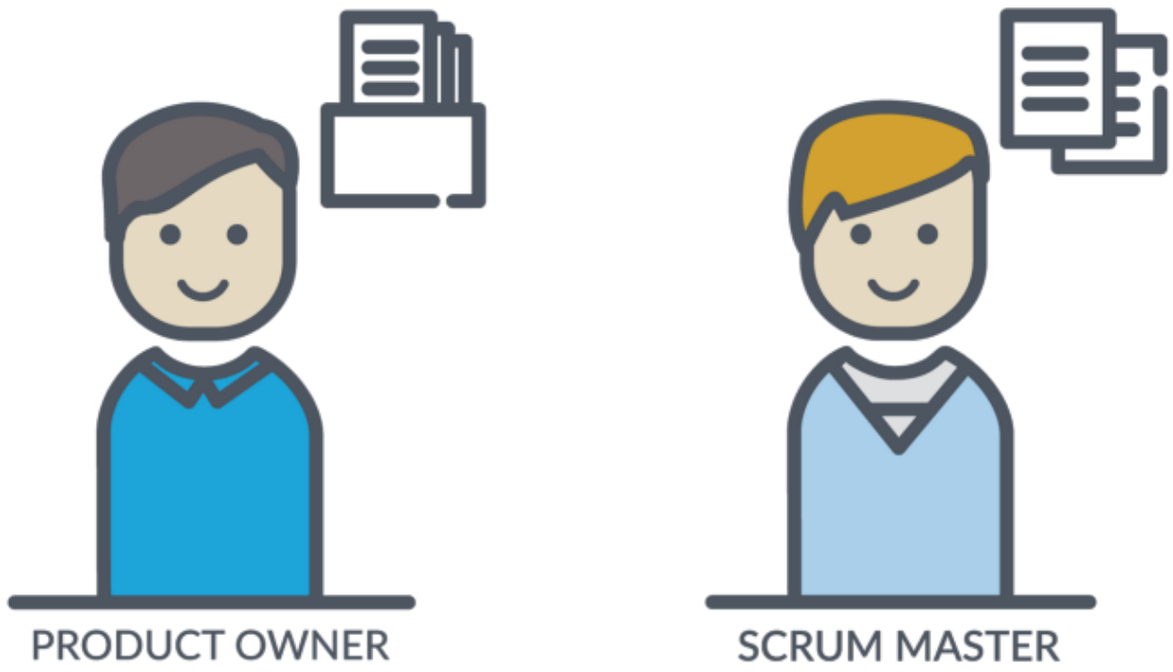
Scrum Overview

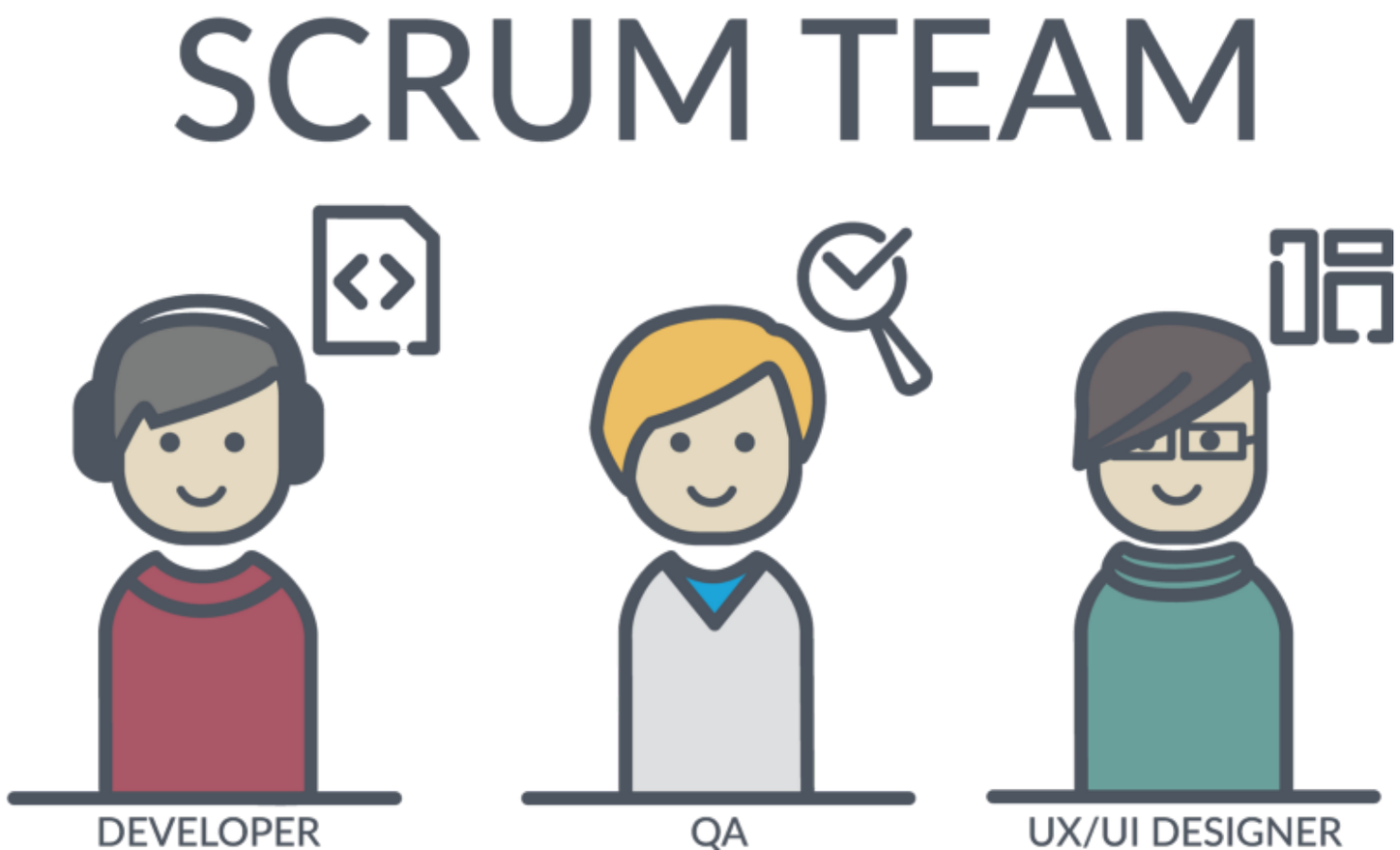
SCRUM FRAMEWORK



Scrum.org

Scrum Rollen





Product Owner: Product Backlog Management

- Product Backlog-Einträge **klar** zu **formulieren**;

- Die Einträge im Product Backlog so zu **sortieren**, dass Ziele und Missionen optimal erreicht werden können;

- Den Wert der Arbeit zu **optimieren**, die das Entwicklungsteam erledigt;

- Das Sicherstellen, dass das Product Backlog **sichtbar**, **transparent** und für alle klar ist sowie zeigt, woran das Scrum Team als nächstes arbeiten wird;

- sicherzustellen, dass das Entwicklungsteam die Product Backlog-Einträge im erforderlichen Maße **versteht**.

Scrum Master: Servant Leader

- Für Verständnis und **Durchführung** von **Scrum** verantwortlich
- **Optimiert** die **Zusammenarbeit**
- Dient dem **Product Owner**
    - Vermittelt Techniken, Verständnis und unterstützt
- Dient dem **Entwicklungsteam**
    - Coaching, Unterstützung, Hindernisbeseitigung
- Dient der **Organisation**
    - Planen, Leiten und Coachen bei der Scrum Einführung

Scrum Entwicklungsteams

- **Selbstorganisierend**
  Niemand sagt dem Team wie aus dem Product Backlog potentiell auslieferbare Funktionalität gemacht wird

- **Interdisziplinär**
  Alle notwendigen Fähigkeiten sind im Team vorhanden

- **Jeder** ist ein “**Entwickler**”
  Scrum kennt keine anderen Titel und Unterteilungen im Entwicklungsteam

- **Das Team als ganzes ist rechenschaftspflichtig**

- 3-8 Personen (“Two Pizza Rule”)

Scrum Artefakte: Product Backlog

- Eine geordnete Listen von allem was im Produkt enthalten sein kann
- Verantwortlich ist der Product Owner
- **Niemals vollständig**, entwickelt sich ständig weiter
- Ein Eintrag enthält eine Beschreibung, die Reihenfolge, die **Schätzung** und den Wert
- Können hierarchisch angeordnet sein und werden erst für einen Sprint verfeinert
- Schätzungen werden vom Entwicklungsteam verantwortet (Endgültige Schätzung erfolg immer von denjenigen, die die Arbeit erledigen werden)

Scrum Artefakte: Sprint Backlog

- Menge der für den Sprint **ausgewählten** Product Backlog-Einträge

- **Plan** für die Lieferung des Produkt-Inkrements und Erfüllung des Sprint Ziels

- Macht die Arbeit **sichtbar**, die das Entwicklungsteam für notwendig erachtet, um das Sprint-Ziel zu erreichen

- Macht Sprint-**Fortschritt** transparent und ermöglicht Nachverfolgung

Scrum Artefakte: Inkrement

- **Ergebnis** eines Sprints inkl. der Inkremente früherer Sprints
- Muss am Ende eines Sprints “**Done**” sein
- Muss im **einsatzfähigen** Zustand sein

Anforderungserhebung

- **Sammeln**
  von Anforderungen der Stakeholder

- **Analysieren**
  Klassifizierung, Sortierung, Bewerten und Vergleichen

- **Spezifizieren**
  Anforderungen eindeutig, testbar und verständlich formulieren

- **Validieren**
  Konsistenzprüfung auf Widerspruchsfreiheit und Sinn

Lastenheft

- Beschreibung der **Benutzeranforderungen**
- Aus **Kundensicht**
- Aussagen in **natürlicher Sprache**
- Sowie Diagramme
- zur **Beschreibung der Dienste**, die das System leisten soll
- Randbedinungen
- **Grundlage** für das Pflichtenheft

Pflichtenheft

- Systembeschreibung aus **technischer** Sicht
- Vom Softwarehersteller zu erstellen
- Detaillierte Festlegung von Funktionen, Diensten und Beschränkungen
- Legt fest **was implementiert** werden soll

Funktionale vs. Nicht-Funktionale Anforderungen

- **Funktionale Anforderungen**
  Eine bereitzustellende Funktion oder ein Service des Systems

- **Nicht-Funktionale Anforderungen**
  Technische Anforderungen, auch “Quality of Service” (kurz QoS): Zuverlässigkeit, Verfügbarkeit, Internationalisierung, Sicherheit, Support

Was sind gute Anforderungen?

- **eindeutig**
  Nur eine mögliche Interpretation

- **vollständig**
  Inkl. aller nicht-funktionalen Anforderungen, explizite Kennzeichnung offener Punkte

- **konsistent**
  Keine Widersprüche

- **korrekt**
  Nur “richtige” Anforderungen enthalten

- **verifizierbar**
  Anhand eindeutiger Messverfahren

- **strukturiert**
  Klare Struktur ohne Redundanz

- **gewichtet**
  Priorisiert bzgl. Wichtigkeit

- **nachvollziehbar**
  Quelle sollte festgehalten werden

[IEEE Std. 830-1998]

Use Cases & User Stories

- Grundlage für das Lastenheft bzw. Product Backlog
- Use Case “**Konto verwalten**”
- User Story “**Konto anlegen**”
  Als *Konto-Verwalter* möchte ich ein neues *Konto* für einen *Kunden* anlegen. Dabei will ich nur den Namen des Kunden eingeben müssen und nach dem Speichern die neue Kontonummer angezeigt bekommen.

- User Story “**Adresse ändern**”
  Als Konto-Verwalter möchte ich bei einem bestehendem Konto die Adresse des Kunden ändern können.

Methoden zur Erhebung von Anforderungen

- **Interview**
  Direkte Kommunikation mit den Stakeholdern

- **Beobachtung**
  Ableitung von Anforderungen aus Prozessbeobachtungen

- **Inventur**
  Vorhandene Unterlagen und Dokumentation dienen als Grundlage

- **Event Storming** (aus DDD)
  Fachexperten und IT-Leute erkunden Domäne mit Post-Its

Aufwandsschätzung

- **Analogie**
  Vergleich mit ähnlichen Projekten und derem Aufwand

- **Function-Point-Verfahren**
  Versucht die Größe eines Systems zu messen (nicht der Umsetzung!)
  Funktionen werden mit Punktwerten versehen
  Misst nur fachlich-funktionale Anforderungen

- **COCOMO** (Constructive Cost Model)
  Primärer Kostenfakter sind Delivered Source Instructions (DSI)
  Nur beschränkt für Aufwandsschätzung geeignet, Codezeilen sind schwer zu schätzen

- **Delphi-Methode**
  Gruppe von Experten werden unabhängig in mehreren Runden mitsamt Feedback befragt