



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Kapitel 05

VOM V-MODELL ZU SCRUM

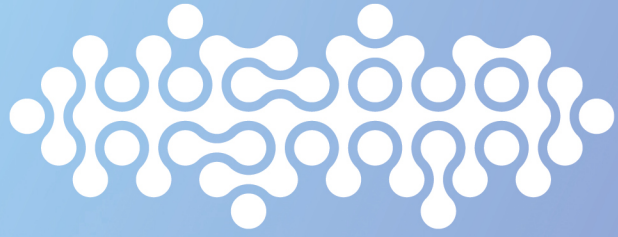
Heutige Agenda

Teil 1: Woran traditionelle Methoden scheitern

- Das V-Modell erklärt
- Übung: "Das Stille-Post-Spiel"
- Das Planungsparadoxon

Teil 2: Scrum in Aktion

- Scrum 3-5-3: Rollen, Events, Artefakte
- Übung: "Das Sprint-Spiel"
- Die Macht des Boards

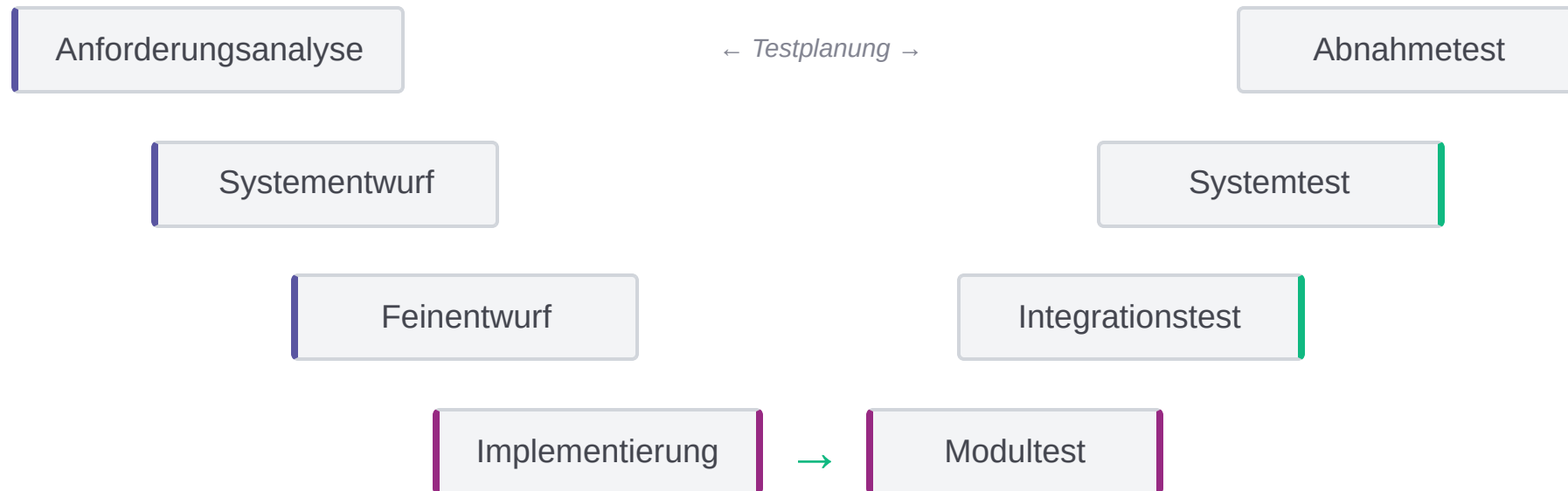


Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Warum traditionelle Methoden scheitern

1

Das V-Modell: Eine kurze Wiederholung



Kernidee: Jede Entwicklungsphase hat eine entsprechende Testphase.

V-Modell: Die Theorie klingt großartig!

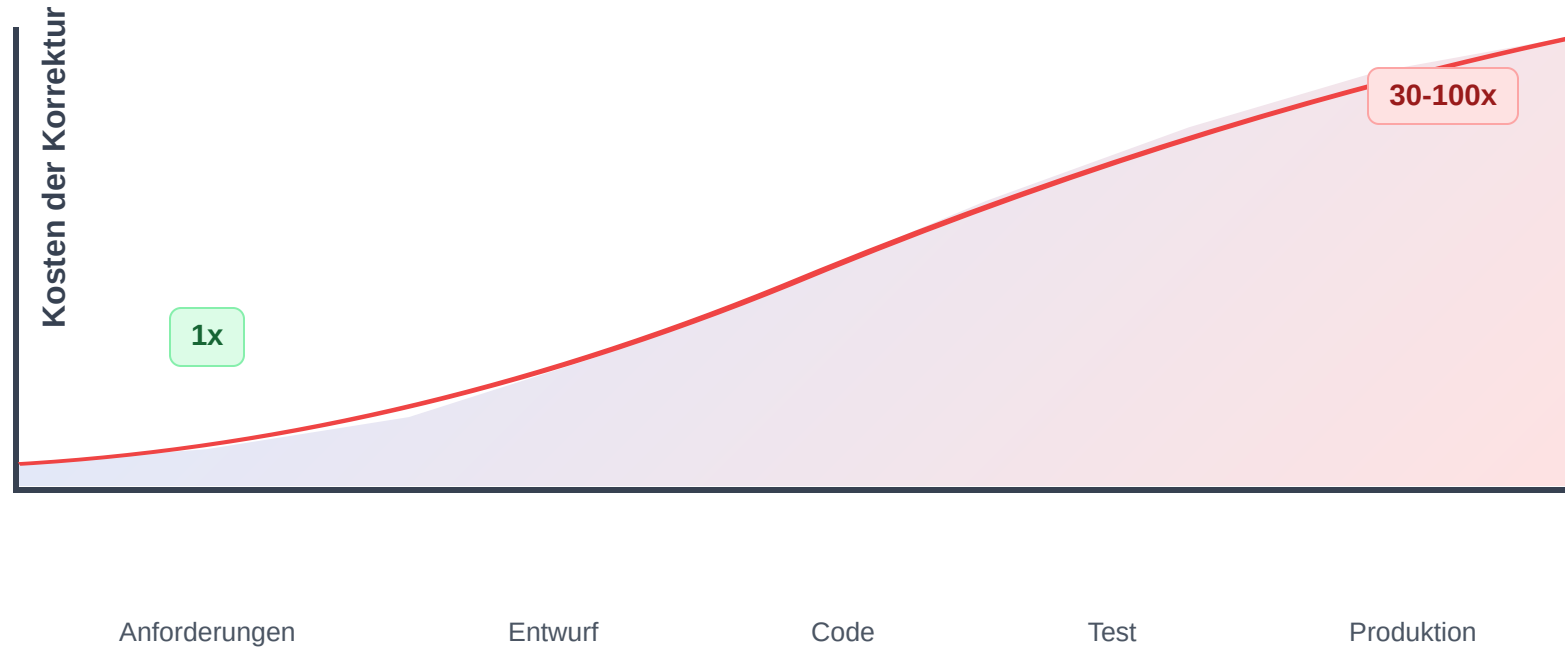
Versprechen:

- Klare Phasen mit definierten Ergebnissen
- Testing von Anfang an geplant
- Rückverfolgbarkeit von Anforderungen zu Tests
- "Wir finden Probleme, bevor sie teuer werden"

Realität:

- Erste funktionierende Software erscheint Monate nach den Anforderungen
- Anforderungen ändern sich, aber der Plan nicht
- Spät gefundene Probleme = teure Korrekturen

Die Kurve der Änderungskosten



Barry Boehm (1976): Einen Bug in der Produktion zu beheben kostet **30-100x** mehr als während der Anforderungsphase.

Die drei falschen Annahmen

Das V-Modell nimmt an:

1. **"Wir können alle Anforderungen im Voraus kennen"**
 - Realität: Nutzer wissen nicht, was sie wollen, bis sie es sehen
2. **"Wir können vollständig entwerfen vor dem Codieren"**
 - Realität: Designprobleme zeigen sich erst beim Bauen
3. **"Wir können es beim ersten Mal richtig bauen"**
 - Realität: Lernen passiert durch Iteration

Das sind nicht nur schwer zu erreichen - sie sind unmöglich.

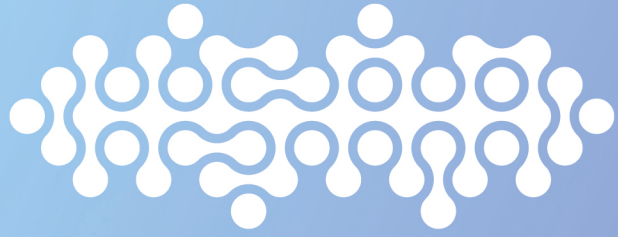
Das Planungsparadoxon

*Wenn wir mehr planen, verschwenden wir Zeit mit Plänen, die sich ändern werden.
Wenn wir weniger planen, bauen wir das Falsche.*

Traditionelle Antwort: Noch SORGFÄLTIGER planen!

- Mehr Dokumentation
- Mehr Review-Meetings
- Mehr Freigaben

Ergebnis: Langsamere Lieferung, gleiche Probleme.



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Übung 1: Das Stille-Post-Spiel

2

Übung 1: Das Stille-Post-Spiel

Ziel: Erleben, warum die Phasentrennung des V-Modells versagt.

 **Zur Web-App beitreten** - Code wird auf dem Bildschirm gezeigt

 **Wichtige Regel:** Keine Fragen zwischen den Phasen!

Thema: Lernraum-Buchungs-App 

Übung 1: Beispiel (Mensa-App)

Runde	Rolle	Ausgabe
1	KUNDE	"Zeige Tagesmenü, Preise, Veggie-Filter und wie lang die Schlange ist"
2	ANALYST	"ANF-1: Menü. ANF-2: Preise. ANF-3: Diät-Filter. ANF-4: Schlangenstatus"
3	ENTWICKLER	"Dropdown, Preisspalte, Checkbox, Ampel für Schlange"
4	TESTER	✗ "wie lang" (Wartezeit) vs "Ampel" (Auslastung)

Was ging schief? Mehrdeutige "Schlange"-Anforderung → verschiedene Interpretationen!

Übung 1: Du bist dran!

Runde	Deine Rolle	Deine Aufgabe
1	KUNDE	Schreibe eine Feature-Anfrage - sei spezifisch!
2	ANALYST	Lies eine fremde Anfrage → schreibe formale Spec
3	ENTWICKLER	Lies eine Spec → beschreibe deine Implementierung
4	TESTER	Vergleiche dein Original mit dem, was gebaut wurde

 **Die App rotiert Dokumente automatisch zwischen den Runden!**

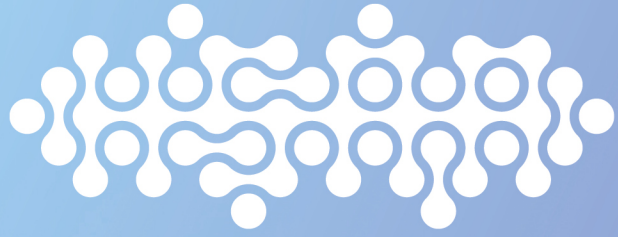
 **Denk dran:** Keine Fragen zwischen den Phasen!

Übung 1: Nachbesprechung

Diskussionsfragen:

1. Wo gingen Informationen verloren?
2. Welche Annahmen hat jede Gruppe gemacht?
3. Wann hast du zuerst bemerkt, dass etwas nicht passt?
4. Wie hätte kontinuierliches Feedback geholfen?

Wichtige Erkenntnis: Spätes Feedback = teure Korrekturen



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Das Scrum-Framework

3

Die agile Antwort

Anstatt Unsicherheit mit mehr Planung zu bekämpfen...

Traditionell	Agil
Komplette Vorausplanung	Kurze Planungszyklen (1-4 Wochen)
Umfangreiche Dokumentation	Funktionierende Software als primäres Artefakt
Change Control Boards	Änderungen als Wettbewerbsvorteil begrüßen
Big-Bang-Releases	Häufige kleine Releases
Testing am Ende	Kontinuierliches Testing durchgehend

Die 3-5-3-Struktur

Scrum bietet Struktur ohne Flexibilität zu opfern.

Die **3-5-3** Struktur:

- **3 Rollen:** Product Owner, Entwickler, Scrum Master
- **5 Events:** Sprint, Planning, Daily, Review, Retrospektive
- **3 Artefakte:** Product Backlog, Sprint Backlog, Inkrement

Die drei Rollen

Rolle	Entscheidet	Entscheidet NICHT
Product Owner	WAS gebaut wird, Priorität	WIE Devs bauen sollen
Entwickler	WIE gebaut wird, Schätzungen	Backlog priorisieren
Scrum Master	Prozess, beseitigt Hindernisse	Tasks zuweisen, Leute managen

Kritisch: Klare Aufgabentrennung!

Product Owner

Die Stimme des Kunden

- Definiert und priorisiert das Product Backlog
- Vertritt die Stakeholder
- Nimmt erledigte Arbeit ab oder lehnt sie ab
- Kommuniziert die Produktvision

Kernprinzip: Eine Person, eine Stimme, eine Priorität.

Kein Komitee!

Entwickler (Development Team)

Die Menschen, die die Arbeit erledigen

- Selbstorganisierend: entscheiden WIE die Arbeit erledigt wird
- Cross-funktional: haben alle benötigten Fähigkeiten
- Liefern jedes Sprint ein potenziell auslieferbares Inkrement
- Typischerweise 3-9 Personen

Kernprinzip: Arbeit wird GEZOGEN, nicht ZUGEWIESEN.

Teammitglieder wählen Aufgaben vom Board!

Scrum Master

Servant-Leader für das Team

- Moderiert Scrum Events
- Beseitigt Hindernisse, die das Team blockieren
- Coacht das Team in Scrum-Praktiken
- Schützt das Team vor Störungen von außen

Kernprinzip: KEIN Projektmanager!

Weist keine Arbeit zu und trifft keine Entscheidungen für das Team.

Schnelles Quiz: Welche Rolle?

Szenario 1: Ein Kunde möchte mitten im Sprint ein neues Feature.

- Wer entscheidet, ob es den Sprint wert ist zu unterbrechen?

Szenario 2: Das Team braucht Server-Zugang, aber die IT reagiert nicht.

- Wer eskaliert dieses Hindernis?

Szenario 3: Das Team debattiert React vs. Vue fürs Frontend.

- Wer trifft die endgültige technische Entscheidung?

Schnelles Quiz: Antworten

Szenario 1: Kunde will neues Feature mitten im Sprint

- **Product Owner** - Sie kontrollieren die Prioritäten

Szenario 2: Team von IT blockiert

- **Scrum Master** - Sie beseitigen Hindernisse

Szenario 3: React vs. Vue Debatte

- **Development Team** - Technische Entscheidungen sind ihre

Die fünf Events

Event	Dauer	Zweck
Sprint	1-4 Wochen	Container für alle Events
Sprint Planning	2-4 Stunden	Arbeit auswählen, Ziel definieren
Daily Scrum	15 Minuten	Synchronisieren, Blocker identifizieren
Sprint Review	1-2 Stunden	Demo für Stakeholder
Sprint Retrospektive	1-2 Stunden	Prozess verbessern

Der Sprint-Zyklus



Das Scrum Board

Warum visuelles Management wichtig ist

ZU TUN	IN ARBEIT	IM REVIEW	FERTIG
US-4	US-3	US-2	US-1
US-5			

Auf einen Blick: Was ist fertig, was ist blockiert, wer braucht Hilfe.

Gesunde vs. Ungesunde Boards

Warnsignal 1: 8 Items "In Arbeit", 0 "Fertig"

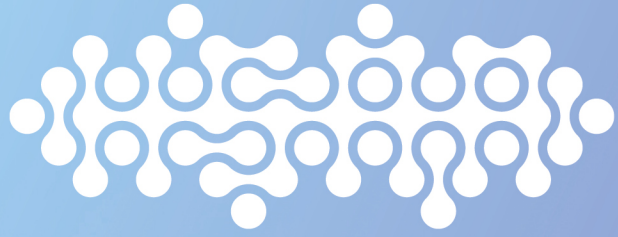
- **Problem:** Zu viel Work in Progress (WIP)
- **Lösung:** Dinge fertigstellen, bevor neue angefangen werden

Warnsignal 2: Board unverändert seit 3 Tagen

- **Problem:** Versteckte Blocker oder keine Updates
- **Lösung:** Daily Scrum, um Probleme aufzudecken

Warnsignal 3: PO verschiebt Karten direkt

- **Problem:** Rollenverletzung
- **Lösung:** Team besitzt seinen Workflow



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Übung 2: Das Sprint-Spiel

4

Übung 2: Das Sprint-Spiel

Ziel: Einen kompletten Scrum-Sprint in 30 Minuten erleben.

Dein Team:

- 1 Product Owner
- 1 Scrum Master
- 3-4 Entwickler

Materialien:

- Product Backlog Karten
- Leeres Sprint Board
- Timer

Übung 2: Sprint-Spiel Regeln

Sprint Planning (5 Min):

- PO präsentiert Top-Stories aus dem Backlog
- Team schätzt und committet sich zum Sprint

Sprint Execution (15 Min):

- Entwickler arbeiten an Aufgaben
- Karten auf dem Board verschieben während du vorankommst
- SM achtet auf Blocker

Sprint Review (5 Min):

- Demo der erledigten Arbeit für den PO

Retrospektive (5 Min):

- Was lief gut? Was kann verbessert werden?

Sprint-Spiel: Der Ablauf

Runde 1 → Runde 2 → Ausführung → Review → Ergebnisse

Runde 1: Backlog-Priorisierung (13 Min)

- 1.1 Wichtigkeits-Voting: Bewerte jede Story ★★★★★
- 1.2 Prioritäts-Voting: Lege Ausführungsreihenfolge fest (0-3)
- 1.3 Finale Priorisierung: PO ordnet neu, Team stimmt Konfidenz ab

Runde 2: Story-Zuweisung (10 Min)

- 2.1 Entwickler-Präferenzen: Wähle Stories, die du willst
- 2.2 PO-Zuweisung: Stories werden Entwicklern zugewiesen
- 2.3 Zufriedenheits-Voting: Bewerte deine Zuweisung 👍 😐 👎

Runde 1: Backlog priorisieren

Phase 1.1: Wichtigkeits-Voting (5 Min)

Alle bewerten die Wichtigkeit jeder Story: ★ bis ★★★★★

Phase 1.2: Prioritäts-Voting (5 Min)

Top-Stories ausgewählt → Stimme über Ausführungsreihenfolge ab (0-3)

Phase 1.3: Finale Priorisierung (3 Min)

- PO kann das Backlog neu ordnen (Drag & Drop)
- Team stimmt Konfidenz ab: 👍 Bereit | 😐 Unsicher | 👎 Bedenken

Runde 2: Stories zuweisen

Phase 2.1: Entwickler-Präferenzen (3 Min)

Wähle die Stories aus, an denen du arbeiten möchtest

Phase 2.2: PO-Zuweisung (5 Min)

PO weist jede Story einem Entwickler zu
(Sieht, wer welche Story wollte)

Phase 2.3: Zufriedenheits-Voting (2 Min)

Bewerte deine Zuweisung: 👍 Zufrieden | 😐 OK | 👎 Unzufrieden
PO kann basierend auf Feedback neu zuweisen

Ausführung & Review

Ausführungsphase

Arbeite an zugewiesenen Stories mit dem Kanban-Board:

Zu Tun → In Arbeit → Im Review → Fertig

Review-Phase

Stories im "Review" brauchen Team-Genehmigung:

- Alle stimmen ab: 👍 Genehmigen oder 👎 Ablehnen
- Alle genehmigen → Story geht nach Fertig
- Ablehnung → Zurück zu In Arbeit für Nachbesserung

Sprint-Spiel: Tipps für Erfolg

- **Stimme ehrlich ab** - Deine Meinung formt den Sprint
- **Kommuniziere** - Nutze das Board, sprich mit Teamkollegen
- **Ziehe Arbeit** - Warte nicht auf Zuweisung
- **Verschiebe Karten** - Halte das Board in Echtzeit aktuell
- **Zeit zählt** - Jede Phase hat einen Timer!

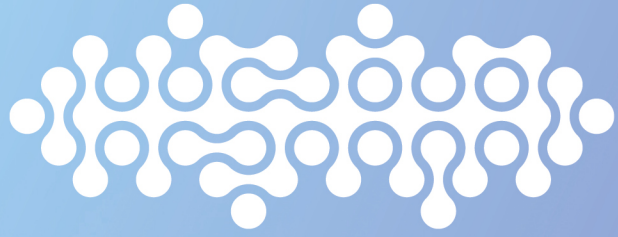
Bereit? Los geht's mit dem Sprint-Spiel!

Übung 2: Nachbesprechung

Diskussionsfragen:

1. Habt ihr alles geschafft, wozu ihr euch committed habt?
2. Was hat euch während des Sprints blockiert?
3. Wie hat das Board geholfen (oder nicht geholfen)?
4. Was würdet ihr im nächsten Sprint anders machen?

Denkt dran: Unvollständige Arbeit kommt zurück ins Backlog!



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Agil in der echten Welt

5

Praxisbeispiel: US-Militär wird agil

März 2024: US Army schreibt Agile für alle Software vor

*"Wir lernen aus aktuellen Konflikten - einschließlich der Ukraine - dass der Erfolg der Army auf zukünftigen Schlachtfeldern von unserer Fähigkeit abhängt, **Software schnell zu aktualisieren.**"*

— Armeeministerin Christine Wormuth

Wenn es für **missionskritische Militärsysteme** funktioniert, kann es für dein Projekt funktionieren.

Traditionelle vs. Agile Militär-Software

Traditionell (5-10 Jahre)	Agil (Wochen-Monate)
Detaillierte Anforderungsdokumente	Kontinuierliche Soldatenbeteiligung
Starrer Beschaffungsprozess	Modulare Verträge
Separate Testphasen	Continuous Integration
Einmal deployen, ewig warten	Kontinuierliche Verbesserung

Die Lektion: "Agile skaliert nicht" ist ein Mythos.

Wichtige Erkenntnisse

1. **V-Modell versagt**, weil seine Annahmen unmöglich zu erfüllen sind
2. **Agile akzeptiert Unsicherheit** und baut kurze Feedback-Schleifen
3. **Scrum bietet Struktur** durch Rollen, Events und Artefakte
4. **Das Board ist kritisch** - Transparenz ermöglicht Zusammenarbeit
5. **Arbeit wird GEZOGEN, nicht zugewiesen** - Teams organisieren sich selbst

Wende das heute an

Für euren Road Profile Viewer:

1. Erstelle ein Product Backlog als GitHub Issues
2. Plane einen 1-Wochen-Sprint mit klarem Ziel
3. Richte ein GitHub Project Board ein
4. Haltet Daily Standups (auch async auf Slack)
5. Macht ein Sprint Review, um euren Fortschritt zu zeigen
6. Führt eine Retrospektive durch, um euch zu verbessern

Der beste Weg, Agile zu lernen, ist es zu praktizieren.

Ressourcen

Offizielle Quellen:

- [Der Scrum Guide](#) - Kostenlos, 13 Seiten
- [US Army Agile Policy \(2024\)](#)

Tools:

- GitHub Projects - Kostenlos, integriert mit Issues
- Jira - Enterprise-tauglich
- Linear - Modern, entwicklerfreundlich



Hochschule Aalen
UNIVERSITY OF APPLIED SCIENCES

Fragen?

DANKE

Als Nächstes: Mache das Scrum-Quiz, um dein Verständnis zu testen!