Übungsblatt 4 – Modelle und agile Software Entwicklung

Luca M. Schmidt

1. Spiralmodell nach Böhm

a. Umgang mit Änderungen und Ansatz

Das Spiralmodell geht **iterativ und risiko gesteuert** mit Änderungen um.

Begründung:

- **Iterativ:** Das Modell arbeitet in Zyklen (Spiralen) mit wiederkehrenden Phasen: Ziele setzen, Risiken analysieren, entwickeln/testen und den nächsten Zyklus planen. Änderungen fließen einfach in neue Iterationen ein.
- **Risikogesteuert:** Jede Iteration enthält eine Risikoanalyse als zentrales Element. Projektänderungen und neue Erkenntnisse werden als Risiken bewertet und in die nächsten Schritte eingeplant.
- **Prototyping:** Frühe Zyklen nutzen Prototypen, um Anforderungen zu klären und Risiken zu reduzieren. Das Feedback führt zu weiteren Anpassungen.
- **Flexibilität:** Anders als bei sequentiellen Modellen können Änderungen problemlos in den nächsten Zyklus integriert werden.

b. Ausprägungen der Kriterien: V-Modell vs. Spiralmodell

Unterscheidungskriterien	V-Modell	Spiralmodell
Ziele und Pläne	Detailliert, frühzeitig und umfassend für das Gesamtprojekt festgelegt.	Pro Iteration definiert/verfeinert; anfangs grob, werden detaillierter.
Risikoanalyse	Nicht expliziter Kernbestandteil des Grundmodells, oft als begleitender Prozess.	Zentraler, expliziter Bestandteil jeder einzelnen Iteration.
Prototypen	Nicht zwingend, aber für Anforderungsanalyse/UI-Design möglich.	Typischerweise in frühen Iterationen zur Risikominimierung & Anforderungsvalidierung.
Simulationen	Möglich, z.B. für Leistungsanalyse, aber nicht Kernbestandteil.	Können Teil der Risikoanalyse oder des Prototypings sein.

Unterscheidungskriterien	V-Modell	Spiralmodell
Tests	Definiert Teststufen (Komponente, Integration, System, Abnahme) parallel zu den Entwicklungsphasen.	Kontinuierlich in jeder Iteration, oft auf Prototypen oder Inkremente bezogen.
Dokumente	Umfassend, formal, meilensteinbasiert, oft hoher initialer Aufwand.	Iterativ erstellt, anfangs weniger detailliert, wächst mit dem Projekt und den Risiken.
Auslieferung	Typischerweise eine Gesamtauslieferung am Projektende.	Inkrementelle Auslieferungen funktionsfähiger Teile sind möglich und oft Ziel.
Inkrementelle Entwicklung	Grundmodell ist sequentiell, nicht inhärent inkrementell.	Von Natur aus inkrementell, da das System in Zyklen erweitert wird.
Umgang mit Änderungen	Schwieriger und teurer, da Pläne früh fixiert; formale Change-Requests.	Flexibler, Änderungen können in nachfolgenden Iterationen eingeplant werden.
Aufwand/Kosten für die Durchführung aller Schritte	Hoher initialer Planungs- und Spezifikationsaufwand.	Anfangs potenziell geringer, kann aber durch viele Iterationen und detaillierte Risikoanalysen auch hoch werden.

---

2. Softwareprozess-Modelle

a. Armbänder zum Zählen der Schritte (Smartphone-Sync)

- **Modellvorschlag:** Agiles Vorgehen (z.B. Scrum oder Kanban) oder inkrementelles Modell
- **Warum:**
  - App und Nutzererfahrung brauchen schnelles Feedback
  - Funktionen können schrittweise entwickelt werden - erst Schrittzählung, dann Schlaftracking usw.
  - Wearable-Markt ändert sich schnell - neue Anforderungen müssen flexibel integriert werden
  - Technische Aspekte wie Bluetooth oder Energiesparfunktionen müssen praktisch erprobt werden

b. Blutdruckmessgerät (Speicher für 100 Messungen, ohne Export)

- **Modellvorschlag:** V-Modell oder Wasserfallmodell
- **Warum:**
  - Anforderungen sind klar und stabil - Blutdruck messen und Werte speichern
  - Medizinische Geräte brauchen Zuverlässigkeit und Genauigkeit
  - Gute Dokumentation für mögliche Zertifizierungen ist wichtig
  - Bei wenigen erwarteten Änderungen lassen sich Aufwand und Zeit besser planen

---

3. Plangesteuert oder agil? (nach Sommerville S. 93/94)

Die Wahl zwischen plangesteuertem und agilem Vorgehen hängt von folgenden Faktoren ab:

Faktor	Tendenz zu plangesteuert	Tendenz zu agil
Spezifikation & Entwurf	Hoher Detaillierungsgrad vor Implementierung nötig	Iterative Entwicklung möglich
Auslieferungsstrategie	Einmalige/seltene Releases	Inkrementelle Auslieferung mit schnellem Kundenfeedback
Team & Größe	Große Teams, verteilte Struktur	Kleine, ko-lokalisierte Teams mit informellem Austausch
Systemkomplexität	Hoher Analysebedarf (z.B. Echtzeitsysteme)	Geringere analytische Komplexität
Systemlebensdauer	Langlebige Systeme mit umfangreicher Dokumentation	Kürzere Lebenszeit oder agile Wartungsstrategie
Entwicklungswerkzeuge	Wenig Unterstützung für Codeanalyse/Visualisierung	Gute Werkzeuge zur Entwurfsbeobachtung
Teamorganisation	Dezentral oder ausgelagert	Zentral mit direkter Kommunikation
Unternehmenskultur	Traditionell, planungsbasiert	Flexibel, anpassungsfähig
Entwicklerqualifikation	Unterschiedliche Qualifikationsstufen möglich	Höheres fachliches Können erforderlich
Regulierung	Externe Genehmigung nötig (z.B. Luftfahrt)	Weniger regulierte Bereiche

In der Praxis (bspw. mein Unternehmen) werden oft Elemente beider Ansätze kombiniert, je nach spezifischen Projektanforderungen.