Software Engineering – Blatt 10

Rasmus Diederichsen Felix Breuninger
{rdiederichse, fbreunin}@uos.de

18. Januar 2015

Aufgabe 10.1: Evolutionäre / inkrementelle Modelle

a) Inkrementelle Modelle

Der inkrementelle Entwicklungsprozess ist durch folgende Aspekte gekennzeichnet:

- Software im Allgemeinen zunächst als Kernprodukt abgeliefert
- Nach und nach werden Features hinzugefügt
- Jedes Feature durchläuft einen kompletten Entwichklungszyklus (z.B. wie im Wasserfallmodell)
- Frühere Releases sind abgespeckte Versionen der späteren.
- Inkremente werden anhand der Benutzung des vorherigen Releases durch den Kunden durchgeplant unt entwickelt.

Vorteile	**Nachteile**
Schneller Weg zur fertigen Kernfunktionalität	Unklar, wann es “fertig” ist
Es werden keine Releases weggeworfen, Qualitätsarbeit in jeder Iteraion	Inkremente v. A. nach User-Feedback → schwer planbar
Entwicklung kann bei Einhaltung on Restriktionen dennoch stets fertiges Produkt liefern (Ressourcenknappheit, Deadlines, Warten auf Technologien, etc.)	
Beliebige Erweiterbarkeit (potenziell langlebig)	
Kurze Iterationen machen Fehler weniger teuer	
Nachträgliche Ergänzungen einfach einzubauen	

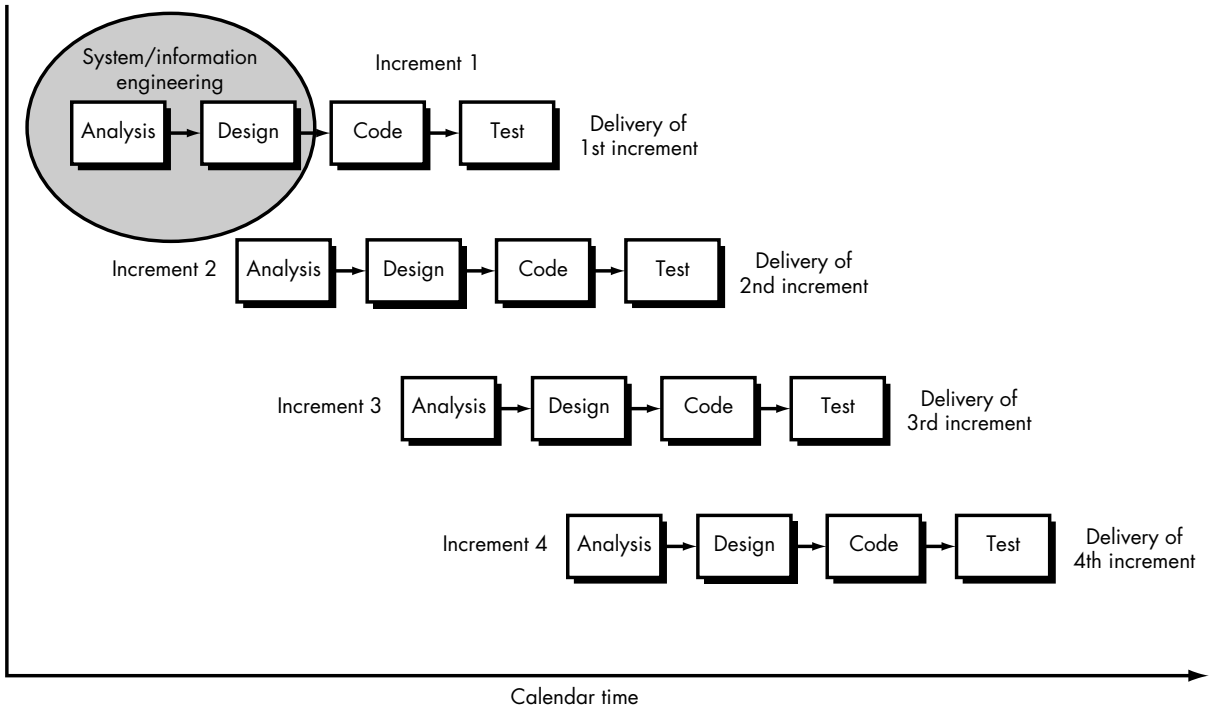

Abbildung 1: Iteratives Vorgehensmodell

b) Evolutionäre Modelle

Das evolutionäre Vorgehen ist charakterisiert durch

- Anforderungen, Pläne, Designs, Aufwandsschätzungen etc. entwickeln sich über die Zeit
- Entwicklung verläuft iterativ, indem die einzelnen Phasen nach Anpassung der Spezifikationen wiederholt werden
- In jeder Iteration werden die Bestandteile der Phasen (s.o.) verfeinert
- im Spiralmodell schreitet die Entwicklung spiralförmig fort, indem aufbauend auf dem Produkt der vorherigen Iteration immer alle Phasen neu durchlaufen werden.
- Anders als beim rein inkrementellen Vorgehen durchlaufen nicht einzelne Features den Entwicklungsprozess und werden “angetackert”, sondern die Gesamtsofware wird überarbeitet.

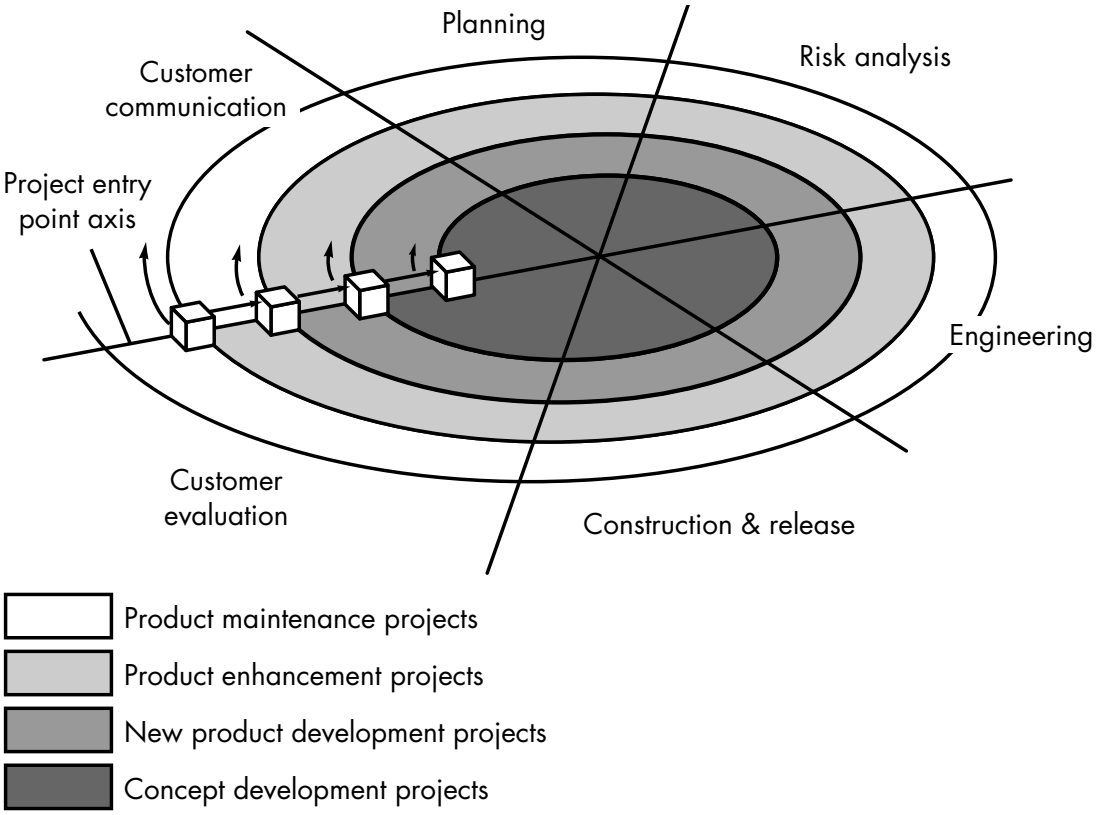

Abbildung 2: Spiralmodell; Evolutionäres Vorgehen mit definierten Task Sets

Vorteile	**Nachteile**
Hohe Anpassbarkeit an Kundenvorstellungen	Schwer planbar, da Anforderungen nicht zu Anfang bekannt
Schneller Prototyp	Versionen werden verworfen
Anforderungen müssen nicht alle zu Beginn bekannt sein	Häufige Änderungen sind schwer zu überblicken und können zu schlechterem Code führen
Sämtliche Projektdokumente werden verbessert, alles ist konsistent	
Prozess hat kein Ende, kann gesamte Lebensdauer abdecken	Kein klares Ende
Zu keinem Zeitpunkt im Lebenszyklus werden Aspekte vernachlässigt.	

Quellen:

1. *Pressman, R.: Software engineering: a practitioner's approach, 5th ed.*
2. *Larman, C.: Agile and Iterative Development: A Manage's Guide*

Aufgabe 10.2: Vorgehensmodelle

1. Da die Anforderungen bekannt sind und ein strikter Zeitplan möglich ist, bietes sich ein lineares Vorgehen wie Wasserfall oder V-Modell an. Es ist nicht erforderlich, möglichst schnell etwas präsentieren zu können, also muss man nicht agil vorgehen.
2. Rapides Prototyping erlaubt, das Problem notdürftig zu hotfixen, der Fix kann dann verbessert und stabilisiert werden. Möglich wäre auch ein Code-and-Fix-Ansatz, da hier höchste Dringlichkeit gegeben ist.
3. Evolutionär vorzugehen, würde es erlauben, die zwangsläufig auftretenden Änderungen miteinzubeziehen.
4. Das V-Modell expliziert die Notwendigkeit und Zuständigkeit verschiedener Tests. Dank der vorliegenden älteren Version kann das Projekt gut von vorn bis hinten durchgeplant werden, bevor man anfängt.
5. Seems legit. Code-and-Fix wäre hier ausreichend, da Dr. Blair kein Softwareingenieur ist und nur flott etwas demonstrieren muss. Die Software wird weder wirklich benötigt, noch wird sie länger in Betrieb sein.

Aufgabe 10.3: Scrum

Aufgabe 10.4: Kanban