eXtreme Programming (XP)

Hubert Baumeister

Softwareentwicklungspraktikum SS 2003

Problem

Kunde kommt mit einer groben Vorstellung seiner Anforderungen zu einem Softwarehaus und möchte wissen wie teuer die Software sein wird und wann die Software fertiggestellt sein wird.

Problem:

- Software kommt zu spät
- Software wird teurer als geplant
- Software erfüllt nicht die Anforderungen des Kunden

Ursachen (Beispiele)

- Unklare Anforderungen
  - Neue Technologien
  - Neue Geschäftsmodelle
  - 30 % der gesamten Entwicklungszeit für Anforderungen um mit 80 % Wahrscheinlichkeit Kosten und Dauer des Projektes abschätzen zu können.
- Planung
  - Wie lange dauert eine Aktivität?

Was ist XP?

- "Leichtgewichtige/agile" Softwareentwicklungsmethodik
- Menge von "extremen" Softwareentwicklungspraktiken
  - Programmieren in Paaren (Code Review)
  - Testen vor dem Implementieren
  - Refactoring (fortlaufender Entwurfsprozeß)
- Entwickelt von Kent Beck und Ward Cunningham vor 5–6 Jahren

Maximen der XP Entwicklung

- Programmiere für die aktuelle Aufgabe
  - Anforderungen können sich ändern
- Schnelle Rückmeldung
  - Kurze Iterationen
  - Ständige Integration

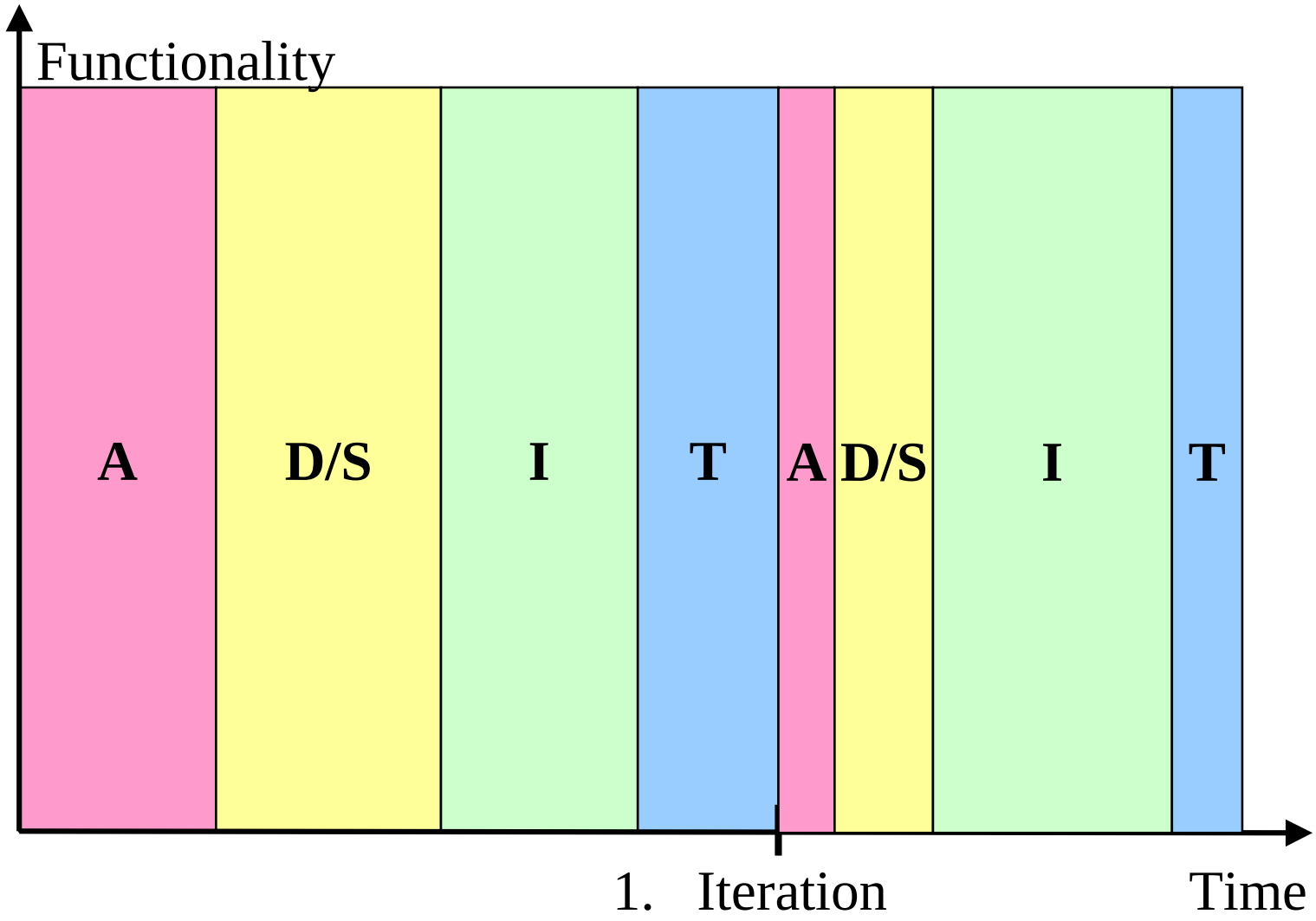
Vorgehensweise XP

1. Kunde definiert User-Stories
2. Planning Game
   (a) Entwickler schätzen Aufwand und Risiko
   (b) Kunde ordnen User-Story Iterationen zu

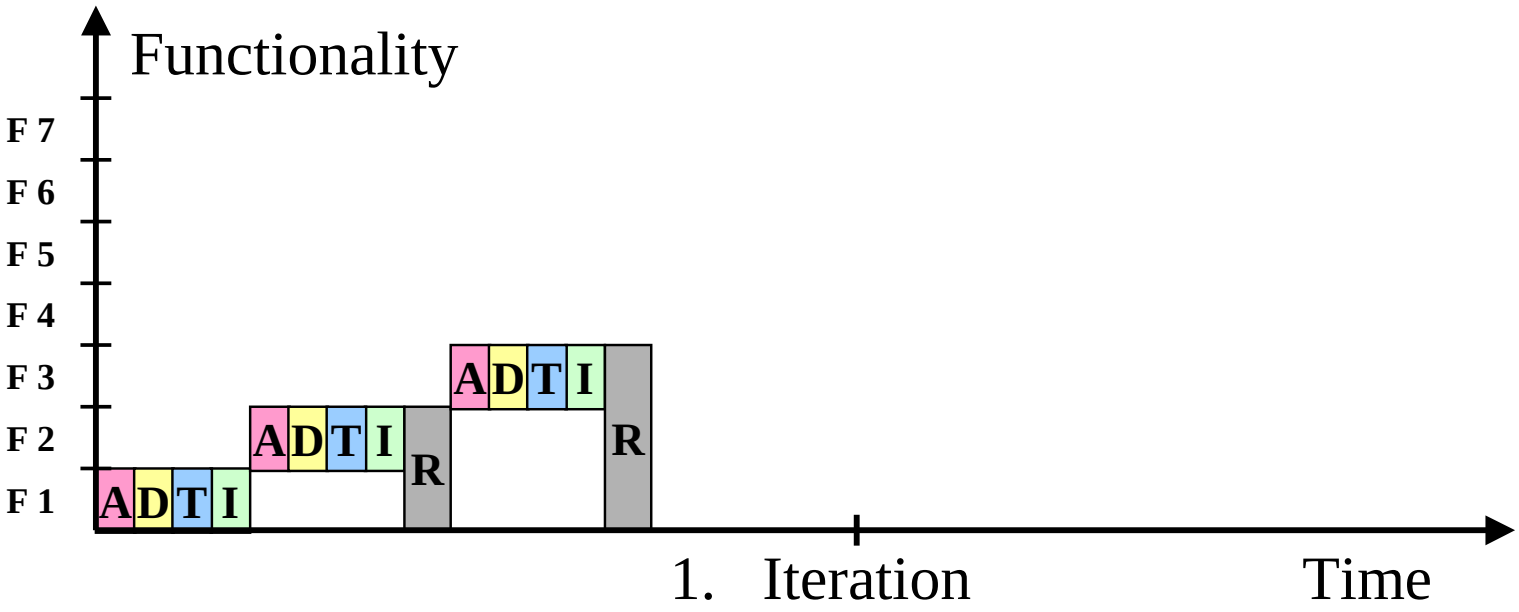
Vorgehensweise XP

1. Pro Iteration
   (a) Entwickler definieren Tasks
   (b) Kunden schreiben Acceptance-Tests
   (c) Folgende Vorgehensweise wird wiederholt, bis Task fertiggestellt ist (Test-Driven Development)
      i. Unit-Test schreiben
      ii. Funktionalität implementieren bis Tests laufen
      iii. Refactor

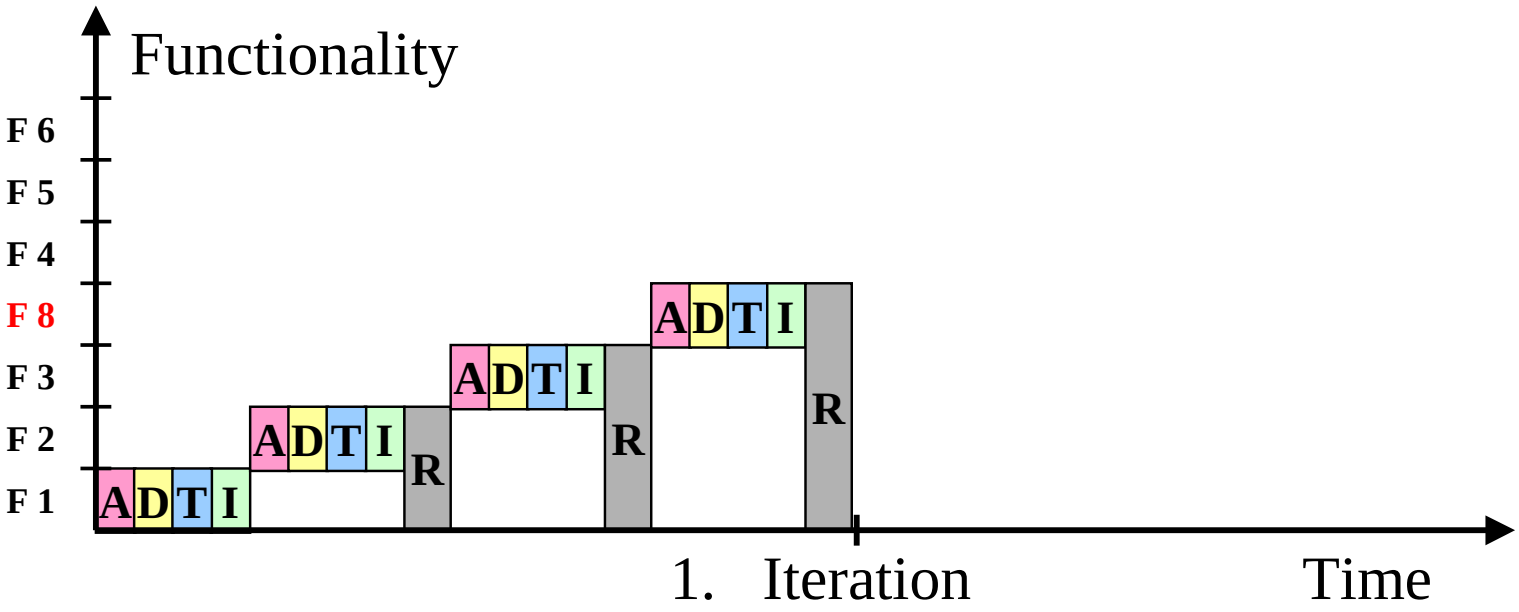
Traditionelle Vorgehensweise



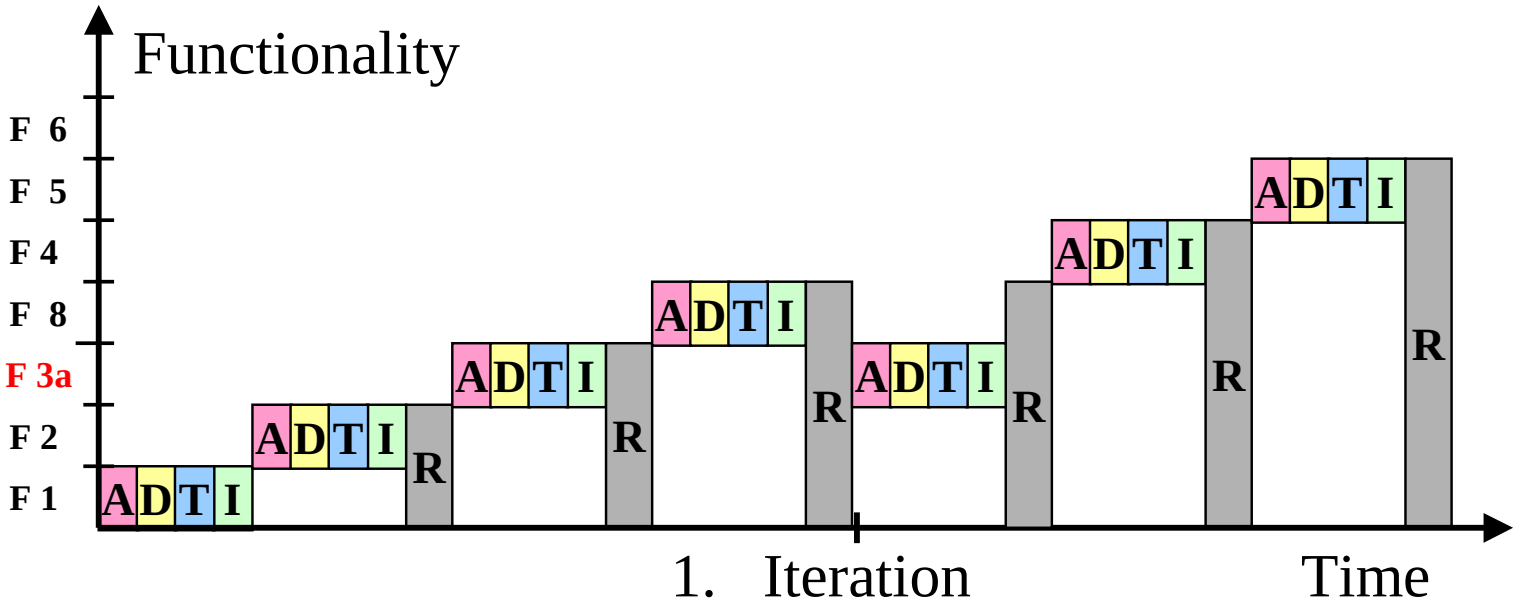
Vorgehensweise XP



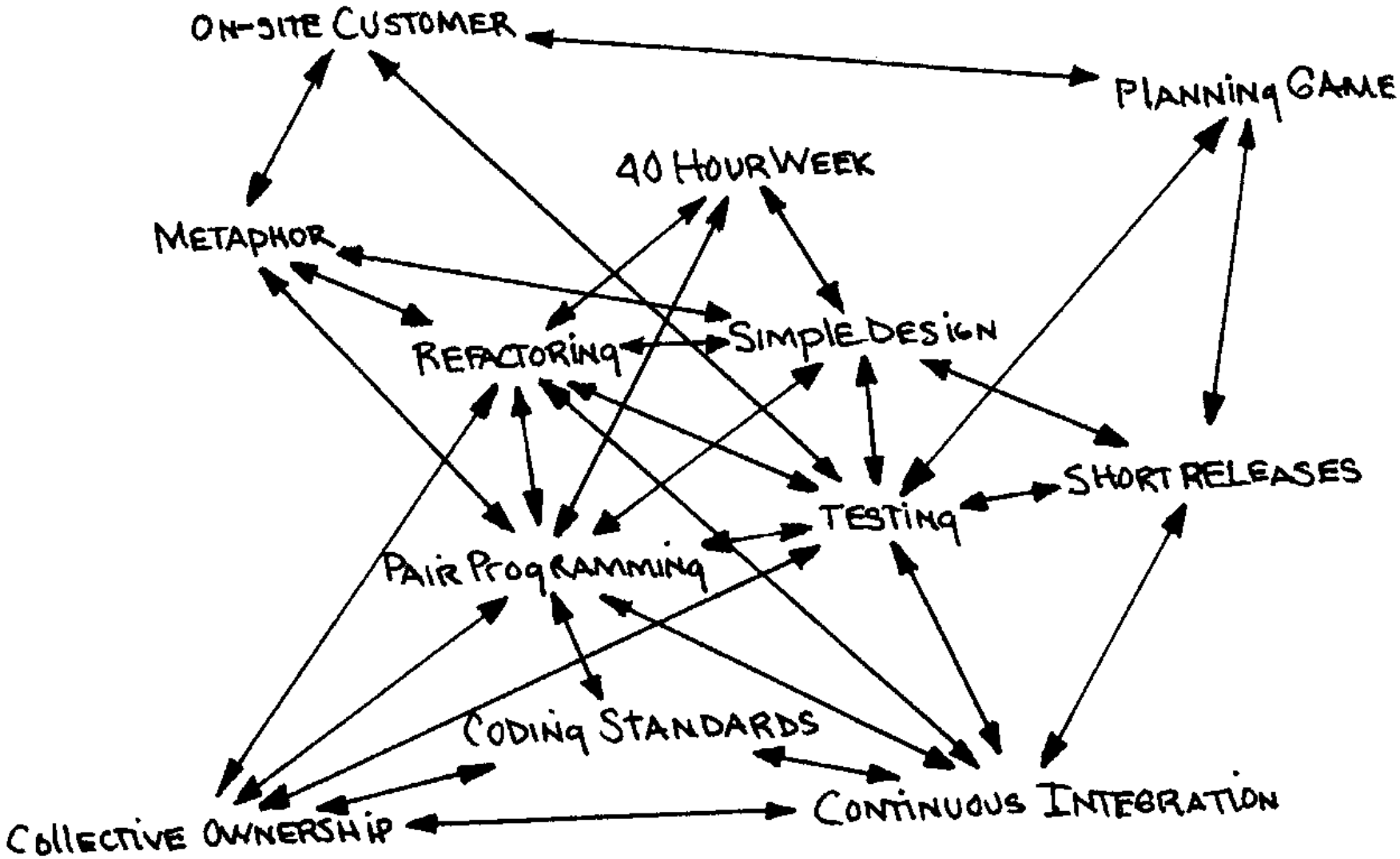
Vorgehensweise XP



Vorgehensweise XP



XP Praktiken



Relevante Praktiken für das SWEP

- Planning Game
- Automatisierte Tests (Unit- und Acceptance Tests)
- Refactoring
- Einfachster Entwurf
- Programmieren in Paaren

Programmieren in Paaren

- Ständiger "Code Review"
- Produktionssoftware wird immer von Paaren programmiert
- Paare sind dynamisch
  - Hilft Wissen zu verbreiten

Einfachster Entwurf

- Keine Verdopplung von Code
- Code vermittelt die Intention des Programmierers
- Keine überflüssigen Klassen / Methoden

Weiteres Vorgehen

- Nächste Woche Planning Game
- Anschließend 3 wöchige Iterationen

Modellierung

- Fähigkeiten zur Modellierung ist in XP wichtig
- Aber es gibt kein definierten Artefakte dafür
- Eine User-Story pro Iteration zur Modellierung:
  - Erlernen von Modellierungstechniken
  - Dokumentation für spätere Weiterentwicklung der Software
  - Dokumentation zum Vergleich mit RUP
  - Verwendung der Modellierung im AGILE Projekt

Literatur: eXtreme Programming

- Kent Beck: *extreme Programming explained.* Addison-Wesley, 1999.
- Kent Beck, Martin Fowler: *Planning Extreme Programming* Addison-Wesley, 2000.
- Martin Fowler: *Refactoring. Improving the Design of Existent Code*, Addison-Wesley
- `http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap`
- `www.XProgramming.com`
- `www.eXtremeProgramming.org`

Literatur: Test-Driven Development

- Johannes Link: *Unit Tests mit Java. Der Test-First-Ansatz.* D-Punkt Verlag, 2002.
- Kent Beck: *Test Driven Development. By Example.*, Addison-Wesley, 2002.
- `www.junit.org`

Literatur: Acceptance Tests

- `www.fitnesse.org`
- `fit.c2.com`