Feature Driven Development

Definition:

The features are small "useful in the eyes of the client" results.

Inhalt

- Key Roles
- Supporting Roles
- Additional Roles
- Prozesse
- Code Ownership
- Meilensteine
- Best Practices
- Literatur

Key Roles: Project Manager

- Administrativer Kopf des Projektes (Budget, Ausstattung des Projekts, Räumlichkeiten, andere Resourcen, ...)
- Einrichten und Erhalten einer optimalen Arbeitsumgebung
- Abschirmung des Entwicklungsteams und Minimierung von Unterbrechungen
- Gibt Fortschritt des Projekts weiter
- Hat das letzte Wort beim Projektrahmen, Zeitplan und Personalausstattung

Key Roles: Chief Architect

- Verantwortlich für Gesamtdesign des Systems
- Verantwortlich für Durchfuehrung von Design-Treffen (Workshops), bei denen das Team am Systemdesign mitarbeitet
- Schlichtet Streit zwischen Chief Programmer's in Designfragen
- Hat das letzte Wort in Designfragen
- Steuert Projekt durch technische Hindernisse hindurch

Key Roles: Development Manager

- Leitet die tägliche Entwicklungsarbeit
- Schlichtet Streit zwischen Chief Programmer's in Resourcenfragen
- Rolle wird oft kombiniert mit Chief Architect oder Project Manager
- Hat das letzte Wort bei Resource-Konflikten
- Steuert Projekt durch potentielle Resourcen-Deadlocks hindurch

Key Roles: Chief Programmer

- Nehmen an der High-Level-Anforderungsanalyse und am Gesamtdesign teil
- Leiten kleine Teams (3 - 6 Leute) durch Low-Level-Analyse, Design und Entwicklung von Features
- Kommunikation mit anderen Chief Programmer's, um alltägliche technische Dinge und Resourcen-Probleme zu lösen

Key Roles: Class Owner

- Mitglieder von kleinen Gruppen (siehe Chief Programmer)
- Design, Implementierung, Test und Dokumentation von Features

Key Roles: Domain Expert

- Typischerweise Benutzer, Kunden, Sponsoren, Analysten
- Verfügen über tiefes Wissen im Problem-/ Anwendungsbereich
- Erläutern den Entwicklern die Aufgaben und korrektes Verhalten des Systems
- "Wissensbasis" für Entwickler
- Wissen der Domain Experts ist kritisch für den Erfolg des Systems

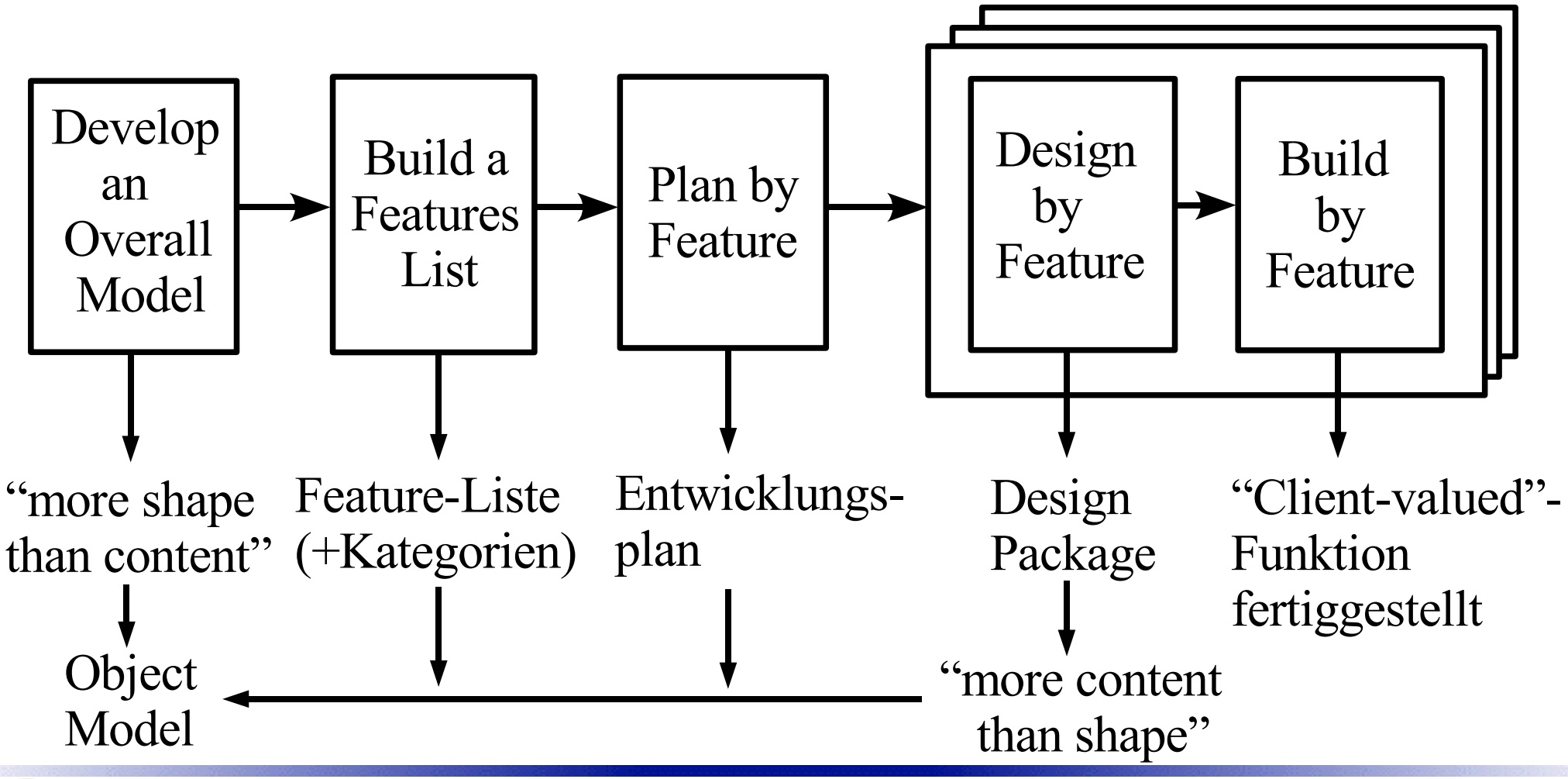
Supporting Roles

- Domain Manager: leitet die Domain Expert's
- Release Manager: überwacht Fortschitt des Projektes
- Language Lawyer / Language Guru
- Build Engineer: Build-Prozess und Versionsverwaltung
- Toolsmith: Entwicklung kleiner Entwicklungswerkzeuge
- System Administrator

Additional Roles

- Tester
- Deployer: Auslieferung des Systems
- Technical Writer: Benutzer- und Online-Dokumentation

Prozesse: Überblick



Develop an Overall Model (1)

- Voraussetzung: Domain Experts, Chief Programmers und Chief Architect wurden ins Projektteam berufen
- Aufgaben:
  - Bildung des Modellierungsteams: Domain Experts, Chief Programmers, Chief Architect, and. Teammitglieder (Rotation)
  - Domain Walkthrough
  - Dokumente studieren
  - Entwurf einer ersten Feature-Liste
  - Erstellung von Teilmodellen und eines Gesamtmodells
  - Notieren von Alternativen

Develop an Overall Model (2)

- Verifikation: Domain Experts nehmen am gesamten Prozess teil
- Ergebnisse:
  - Objektmodell mit Klassen- und Sequenzdiagrammen
  - Informale Feature-Liste
  - Notizen

Build a Features List (1)

- Voraussetzung: Gesamtmodell wurde erstellt
- Aufgaben:
  - Bildung des Teams zur Erstellung der Feature-Liste: vgl. Erstellung des Gesamtmodells
  - Erstellen der Feature-Liste
  - Setzen der Prioritäten:

    must have / nice to have / add if we can / future
  - Aufteilen komplexerer Features: Ziel ist Implementierungsdauer von max. 2 Wochen

Build a Features List (2)

- Verifikation: Domain Experts nehmen am gesamten Prozess teil
- Ergebnisse: Feature-Liste...
  - vollständig,
  - gegliedert, z.B. Major Feature Sets (Bereiche), Feature Sets (Aktivitäten), Features (Schritte innerhalb der Aktivität),
  - geprüft.

Plan by Feature (1)

- Voraussetzung: Feature-Liste wurde erstellt
- Aufgaben:
  - Bildung des Planungsteams: Project Manager, Development Manager, Chief Programmers, Chief Architect
  - Bestimmung der Reihenfolge
  - Zuweisung der Verantwortlichkeiten für Features und Feature Sets an Chief Programmers
  - Zuweisung der Klassenverantwortlichkeiten an Entwickler

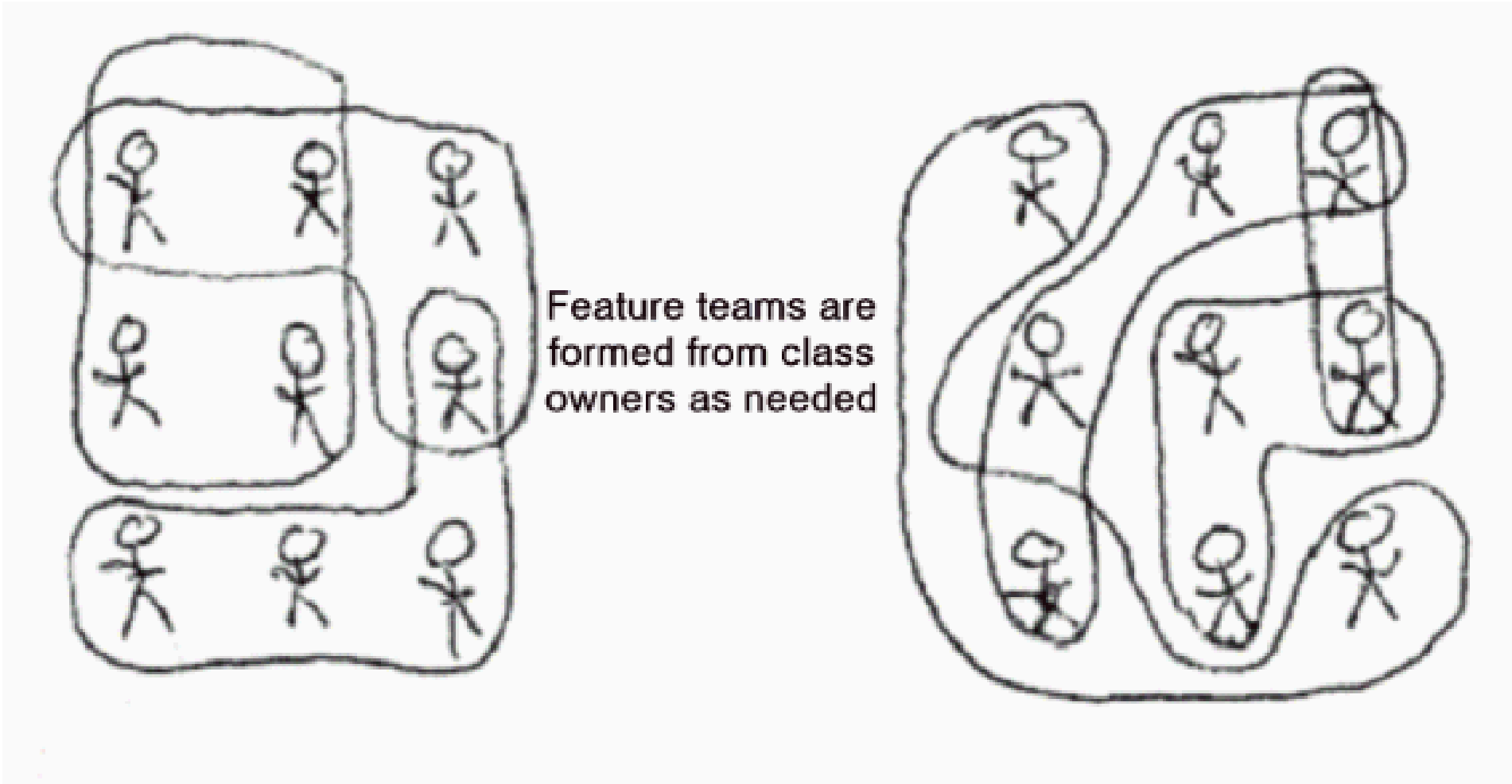
Plan by Feature (2)

- Verifikation: Self-Assessment
- Ergebnisse: Entwicklungsplan mit
  - Feature Sets mit Datum der Fertigstellung (Monat/Jahr)
  - Zuordnung der Chief Programmers zu den Feature Sets
  - Liste der Klassen und deren Class Owners

Class Ownership (1)

- Class Ownership beschreibt, **wer** für den Code einer Klasse verantwortlich ist
- mehrere Eigentümer sind möglich
- erstellte Sequenz-Diagramme zeigen, welche Class Owners zu einem Feature gehören

Class Ownership (2)



Design by Feature (1)

- Voraussetzung: Planung abgeschlossen
- Aufgaben:
  - Bildung des Feature-Teams: Chief Programmer, Class Owners
  - Domain Walkthrough mit Domain Expert (optional)
  - Dokumente studieren (optional)
  - Erstellung von Sequenz-Diagrammen
  - Schnittstellen festlegen (Methoden festlegen, Definition der Parameter und Rückgabewerte)
  - Object Model überarbeiten

Design by Feature (2)

- Verifikation: Design-Inspektion
- Ergebnisse: Design Package mit
  - Design-Dokument
  - Liste der referenzierten Anforderungen
  - Sequenzdiagramme
  - Design-Alternativen (soweit vorhanden)
  - geändertes Object Model
  - javadoc

Build by Feature (1)

- Voraussetzung: Design by Feature abgeschlossen
- Aufgaben:
  - Implementierung
  - Code-Inspektion
  - Unit Tests
  - Code-Freigabe, Commit zur Versionsverwaltung

Build by Feature (2)

- Verifikation:
  - Code-Inspektion,
  - Unit Tests
- Ergebnisse: Abschluss der Feature-Entwicklung
  - neue und geaenderte Klassen wurden inspiziert und getestet
  - Einchecken des Source-Codes

Meilensteine

- Design by Feature
  - Domain Walkthrough
  - Design
  - Design-Inspektion
- Build by Feature
  - Implementierung
  - Code-Inspektion
  - Einchecken des Source-Codes

Best Practices

- Reporting / Sichtbarkeit der Ergebnisse
- Einchecken des Codes erst nach Code-Inspektion

  Folgerung: automatisches Backup der Entwicklungsumgebung!
- Regelmäßige Builds: wöchentlich, täglich oder noch kürzer
- Versionsverwaltung für
  - Code,
  - Dokumentation,
  - Tests,
  - Skripte und andere Hilfsmittel.

Template (1)

<Beschreibung> Stefan Diener, 04.01.2007

Arbeitspaket <x.y.z>

<Beschreibung>

1 Feature Team

Chief Programmer: <Name>
Feature Team: <Name>, <Name>, <Name>

1.1 Anwendungsfälle (Use Cases)

- <Liste>

1.2 Detaillierte Anwendungsfälle

- <Liste>

1.3 Anforderungen

1. <Punkt #1>
2. <Punkt #2>

1.4 Test-Anforderungen

1. <Punkt #1>
2. <Punkt #2>

Template (2)

1.5 Zeitplanung

Feature		AP <x.y.z>
Bereich		<Titel des Arbeitsbereichs>
Beschreibung		<Beschreibung>
Walk-through	Geplant	
	Ist	
Design	Geplant	
	Ist	
Design Review	Geplant	
	Ist	
Implementierung	Geplant	
	Ist	
Code Review	Geplant	
	Ist	
Commit	Geplant	
	Ist	
Doku	Geplant	
	Ist	

Template (3)

1.6 Beteiligte Klassen und Dateien

Klasse / Datei	Besitzer	CVS Revision
Package/Verzeichnis:		

Template (4)

2 Design

2.1 Walk-through

Domain-Experte: <Name>

2.2 Referenzierte Dokumente

[1]<Liste>

2.3 Sequenz-Diagramme / Andere Diagramme

- <Liste>

2.4 Änderungen am Gesamtmodell

- <Liste>

2.5 Änderungen an Klassen und Methoden

- <Liste>

2.6 Design Review

Datum	Bereich	Reviewer	Protokoll	Status

2.7 Zeitabschätzung

Zeit bis zum Design Review	
Geschätzte restliche Zeit	
Geschätzte Gesamtzeit	
Verifikation des Projektplans	❑

Template (5)

3 Implementierung

3.1 Hinweise

Datum	Status

3.2 Unit Tests

- <Liste>

3.3 Check-Liste

- cvs checkout/update: <Datum>
- ant test: <Datum>
- ant javadoc: <Datum>
- ant pmd: <Datum>
- Windows Test: <Datum>, <JVM Version>
- Linux Test: <Datum>, <JVM Version>

3.4 Code Review

Klasse	Reviewer	Protokoll	Status

Template (6)

4 Commit

Eingecheckt in Applikations-Version <X.Y.Z>.

Check-Liste:

- Alle modifizierten Dateien eingecheckt?
- Alle neu erzeugten Dateien eingecheckt?
- Alle modifizierten / neuen Test-Dateien eingecheckt?
- Alle Projekte im Workspace überprüft?

Template (7)

5 Dokumentation

Dokument	Abschnitt	Seite	Reviewer	Protokoll	Status
Release Notes	-	-			
Handbuch					

Template (8)

6 Arbeitspaket-Tagebuch

Datum	Kommentar

Weiterführende Literatur

- Palmer, Felsing: A Practical Guide to Feature-Driven Development (Prentice Hall)
- Palmer: Feature Driven Development and Extreme Programming (The Coad Letter: Modeling and Design Edition, Issue 70): http://dn.codegear.com/article/29684
- http://www.featuredrivendevelopment.com/