Softwareengeneering

Was ist Softwareengeneering und warum sollte man es nutzen?

Def.:

Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.

Planung und Analyse

Anforderungserhebung:

(funktionale und nichtfunktionale Anforderungen, ...)

Aufwandsschätzung:

(Personal, Hardware, verfügbare Zeit, Machbarkeit, ...)

Wahl des Entwicklungsmodells

(Wasserfallmodell, Spiralmodell,...)

Entwurf und Entwicklung

Wahl der Softwarearchitektur:

(Datenflussnetz, Objektnetz, Ablagebasiertes System, ...)

Modularisierung:

Aufteilung in (wiederverwertbarer) Module

Schnittstellendefinition:

(Rechtemanagement, Code Koventionen)

Validierung und QA

Testen:

(Modultests, Systemtests, Akzeptanztests,...)

Problemmanagement:

(Reaktion auf Fehler und Fehlern vorbeugen)

Statische Analyse:

Fehler vor dem Ausführen von Code finden

Dokumentation:

(kommentierter Quellcode, Handbuch, Verfahrensdokumentation, ...)

Prozessmodell

Wir bauen ein Schloss

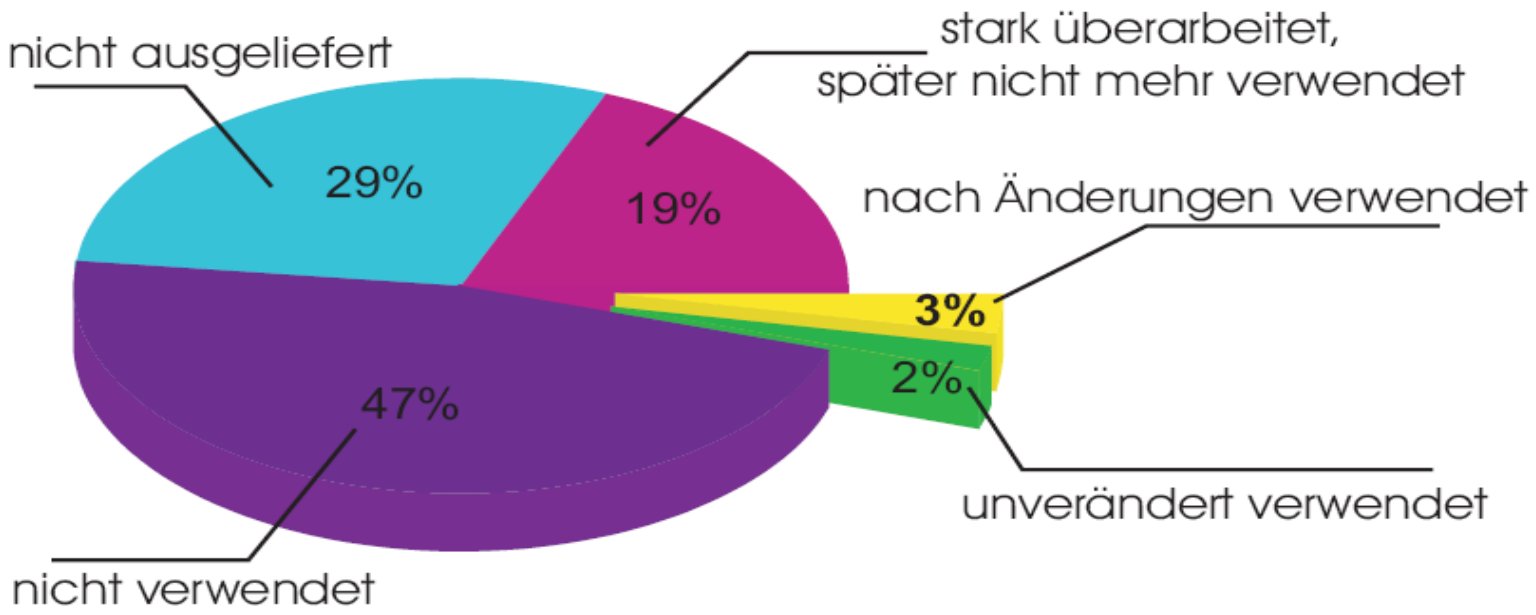
Genauer Architektenentwurf
Bauplan
Fundamant
Wände
Dach

Wasserfallmodell
Rational Rose

Hundehütte
Fenster (neue Wand?)
Häuschen
Haus (neues Fundament?)
Schloss

Rapid Prototyping
Agil
Extreme Programming

Gibt es ein Problem bei der Software - Entwicklung?



Quelle: Financial Times 23.01.89, bezieht sich auf Software-Projekte der US-Regierung

Detaillierte Planung

Vorteile (Wenn man es kann!)

- Effizient
- Zahlreiche Werkzeuge zur Prozesssteuerung
- Bekannte, getestete Verfahren
- Zusammenarbeit in sehr großen Teams möglich

aber

Ist Softwareentwicklung planbar?
Je mehr du nach Plan arbeitest, desto mehr bekommst du, was du geplant hast, aber nicht, was du brauchst.

Rapid Prototyping

Vorteile

- Immer eine lauffähige Version vorhanden
- Anforderungen müssen nicht von vorn herein genau verstanden werden, können laufend präzisiert werden
- Wechselwirkungen zwischen Komponenten werden frühzeitig erkannt
- Fertigstellungsgrad besser verifizierbar
- Qualitätssicherung kann frühzeitig eingebunden werden
- Flexibilität bei unerwarteten Problemen
- Motivation der Mitarbeiter

Nachteile

- Vieles muss mehrfach erarbeitet werden
- Verführt zu mangelhafter Dokumentation, unklarer Aufgabenstellung
- Doppelte Arbeit
- Geringe „äußere“ Disziplin erfordert viel „innere“ Disziplin

Beispiele für Agile Methoden:

- Auf genaue Spezifikation zu Beginn des Projektes verzichten
- Story-Cards
- Nur notwendige Dokumentation
- Testgetriebene Entwicklung
- Ständige Refaktorisierungen

Zusammenarbeit bei Extreme Programming

- Tägliche Besprechungen im Gesamtteam
- Paarprogrammierung
- Wechselnde Gruppen
- Jeder soll Gesamtübersicht haben
- Keine Überstunden!