TB 2 - Softwaresysteme

1. Vorgehensmodelle

Als Grundlage SDLC

SDLC

Software Development Life Cycle

Gute Planung führt zu geringeren Betriebs- & Wartungskosten

Was 1. Idee & Projektanstoß 2. IST-Erhebung - was gibts bereits? - wie sehen die Systeme aktuell aus? 3. Anforderungen erfassen - **funktionale** (was soll SW können?) & **nicht-funktionale** (Performance, Sicherheit, . . .)

Wie 4. System & Kompnentenentwurf - Systemarchitektur - technische Spezifikation - Schnittstellen - . . .

Implementierung 5. Impelentierung 6. Komponententests 7. Integrations & Systemtests 8. Abnahmetests (hält System Spezifikation)

Betriebnahme - Deployment/Release - Außerbetriebnahme alter Systeme - Betrieb & Wartung

Phasenmodelle

Entwicklung verläuft sequentiell & schrittweise

Beispiele: - Wasserfallmodell - Spiralmodell

Wasserfallmodell

- alte Herangehensweise
- kommt aus anderen Disziplinen & klassischen Projekten
- eher weniger bei SW-Projekten
- wenn eine Phase abgeschlossen ist gibt es kein zurück mehr
- erst wenn vorherige abgeschlossen ist kann nächste Phase beginnen
- Kosten bei fixen Anforderungen leicht abschätzbar
- **Dokumentgetrieben** (nach jeder Phase muss ein Dokument vorliegen)
- **top-down**
- nicht mehr anwendbar bei SW (Anforderungen können sich schnell änden)
- Planungsfehler erst spät ersichtlich (late design breakage)

Spiralmodell

- es gibt Phasen die sich wiederholen
- Es wird in Zyklen gedacht

Schritte: 1. Ziele definieren für nächsten Zyklus 2. Risikoanalyse & Prototyping 3. Durchführung und Evaluation 4. Planung der nächsten Phase

- frühzeitige Evaluierung
- Prototypische umsetzung
- Risikominimierung

V-Modell

- verfolgt Test Driven Development
- Dokumentorientiert
- Fokus auf Qualitätssicherung

linke Seite - Etappen des SDLC - vor Durchführung Tests ausdenken & Implementierung Evaluieren

rechte Seite - Tests für jew. Entwicklungsschritte - auf technischer Ebene = funktioniert System überhaupt? - auf benutzer Ebene = Bieter das System dem Nutzer den gewünschten Nutzen - utility/waranty

RUP - Rational Unified Process

- erster Schritt in Richtung agile Modelle
- basiert auf UML (beschreibt auf allen Ebenen Projekt mit Hilfe von UML-Diagrammen => Ausgehend von UseCases)
- Architekturzentriert
- in jeder Phase werden Workflows durchlaufen
  - Business Modelling
  - Requiring (Anforderungen erheben)
  - Analysis & Design (Grobspezifikation)
  - Implementation
  - Tests
  - Deployment
- Supporting Workflows
  - Configuration & Change Management = wie reagiert man auf Anforderungsänderungen
  - Project Management
  - Environment = Arbeitsumgebung schaffen
- Aufwand für jeden Workflow ist abhängig von der aktuellen Phase
- in jeder Phase kann es 1 bis meherer Iterationen geben die jew. ein Produktinkrement liefern
- Elaboration braucht am meisten Aufwand & Zeit

- Late Design Breakage ist sehr unwahrscheinlich

Phasen: 1. Inception - Anforderungen identifizieren - Wirtschaftlichkeit - Risikoanalyse - Machbarkeitsprüfung - Validierung mittels ersten Prototypen - **LCO** = Lifecycle Objective Milestone 2. Elaboration - Architektur erstellen - technische Spezifikation) = Lifecycle Architecture Milestone (= Point of no return) 3. Construction - Umsetzung/Implementierugn - **Initial Operational Capability Milestone** = fertiges System 4. Transition - Übernahme von Entwicklungs- auf Produktionsumgebung - Testen - Inbetriebnahme - **Product Release**

Agile Modelle

- Anforderungen sind veränderlich (daher sind Kosten schwer einschätzbar)
-