SEIP

Architekturprinzipien betriebliche Informationssysteme (Quasar)

Architektur: Arbeitsteilung, erst entwerfen Architekten den Architekturplan, dieser wird dann von anderen verfeinert

Architektur hat verschiedene Sichten: physikalische Sicht (Grundriss, Aufriss), logische Sicht (Netzwerkplan, Abwasserplan, Statik), Detailgrad

Ablauf: 1-2 Architekten entwerfen Architektur mit (nichtfunktionalen) Zielen und geben den Rahmen für die Entwicklung vor. 3-4 Spezifikateure entwerfen Detailpläne.

Ziele der Architektur:

- Beherrschbare Teilstrukturen finden
- Entwicklung mit mehreren Teams
- Entwicklung in Stufen
- Hohe Verständlichkeit
- System muss änderbar sein
- Wartbarkeit
- Wiedererkennbare Lösungen
- Einheitliche, konsistente Lösungen
- Regeln, Konventionen
- Infrastruktur berücksichtigen
- Austausch/Upgrade von Teilsystemen

Merkmale einer Komponente: Export-Schnittstelle, Import-Schnittstelle, versteckt die Implementierung, Einheit der Wiederverwendung, Zusammengesetzt aus verschiedenen Komponenten, wesentliche Einheit des Entwurfs.

Arten von Schnittstellen: Benutzerschnittstelle, Programmierschnittstelle

Eine Schnittstelle fasst Operationen zuasmmen und besteht aus: einem Namen, einer Menge von Operationen, einem Protokoll im Sinne von Reihenfolgen und Restriktionen beim Aufruf.

Architekturprinzipien: Abstraktion, Modularisierung, Kapselung, Hierarchische Dekomposition, Separation-of-Concerns, Einheitlichkeit

Beispiele für Software-Architekturstile: Schichtenarchitektur (Datenhaltung, Logik, Präsentation), Model-View-Controller, Pipes & Filters, Regelbasiertes System, Blackboard Architektur, Service Orientierte Architektur, Event driven, Reflection, Interpreter.

"Software, die sich unterschiedlich schnell ändert, wird in unterschiedliche Module aufgeteilt." [Parnas, 1972]

Fachliche Änderungen (Neue Beitragssätze zur Sozialversicherung)
vs.
Technische Änderungen (Neue Java-Version, …)

Software Kategorien:

- 0-Software (unabhängig von Anwendung und Technik, Ideal wiederverwendbar)
  z.B. Klassenbibliothek für Strings und Behälter
- A-Software (Bestimmt durch die fachliche Anwendung, Unabhängigkeit von Technik) meist größter Teil,
  z.B. Mitarbeiter, Buchung
- T-Software (Unabhängig von der fahclichen Anwendung, Experte für eine technische Komponente, dafür wiederverwendbar)
  z.B. Zugriffsschicht auf Datenbank
- R-Software (Reine Transformation = Repräsentation, Mischung aus A und T)
  z.B. Bildschirmdarstellung in XML umsetzen
- AT-Software (Vermischt von Technik und Anwendung, Grundsätzlich zu vermeiden, da schwer zu warten) Wiederverwendung nahezu ausgeschlossen

Blutgruppen-Kalkül: A+0 = A, T+0 = T, A+T = AT

Quasar-Software-Kategorien liefern eine A/T-Trennung auf Architekturebene

- Technische Infrastruktur (TI-Architektur)
- Technische Architektur (T-Architektur) verbindet A und TI-Architektur
- Anwendungsarchitektur (A-Architektur)

Testing in industriellen Software-Projekten

Qualitätsmerkmale: Funktionalität, Zuverlässlichkeit, Benutzbarkeit, Effizienz, Änderbarkeit, Übertragbarkeit

Konstruktive QS-Maßnahmen zur Vermeidung von Fehlern:

- Produktorientierte Maßnahmen: Templates für Dokumente (z.B. Pflichtenheft), Verwendung von Modellierungssprachen wie UML, Styleguide für Programmierung und Benuteroberfläche.
- Prozessorientierte Maßnahmen: Verwendung eines durchgängigen Vorgehensmodells, Regeln und Werkzeuge, Kommunikations- und Meeting-Kultur, Peer-Reviews

V-Modell:

Anforderungsentwicklung <-> Abnahmetest
Analyse <-> Systemtest
Entwurf <-> Integrationstest
Komponententest
Implementierung

Testfälle werden aus den Qualitätsanforderungen abgeleitet zur Verifizierung der Eigenschaften des IT-Systems

Ad-Hoc-Test: Intuitives Testen kann zusätzlich gemacht werden, kann aber systematische Tests nicht ersetzen.

Testarten: Funktionale Tests, Nicht-Funktionale-Tests, Fehlertests, Datenkonsistenztests, Wiederinbetriebnahmetests, Interoperabilitätstests, Installationstests, Oberflächentests, Stresstests, Lasttests, Performance-Tests, Sicherheitstests.

Teststufen: Komponententest (Entwickler), Integrationstest der Schnittstellen (Tester), Systemtest des Gesamtsystems (Tester), Abnahmetest (Kunde)

Wirtschaftlichkeit von IT-Projekten

$$Wirtschaftlichkeit\ durch\ IT = \frac{Nutzen\ durch\ IT}{IT - Kosten}$$

Effizienz: Gleicher Nutzen mit geringerem Aufwand
Effektivität: Gleicher Aufwand mit größerem Nutzen

Business-Case-Analyse:

- Wer unterstützt das Projekt? Wo sind Widerstände zu erwarten?
- Was ist das Ziel des BC?
- Welche inhaltlichen und zeitlichen Abhängigkeiten bestehen?
- Kostenanalyse (Investitions-, Betriebs-, Prozesskosten (Personalkosten))
- Nutzenanalyse
- Risikoanalyse

Kapitalwertmethode: $K = \sum_{t=0}^{n}(E_t - A_t)\frac{1}{(1+i)^t}$

K = Kapitalwert
Et = Einzahlungen am Ende der Periode t (Nutzen)
At = Auszahlungen am Ende der Periode t (Kosten)
i = Kalkulationszinsfuß
t = Periode (t = 0, 1, 2 ..., n)
n = Nutzungsdauer des Investitionsobjektes

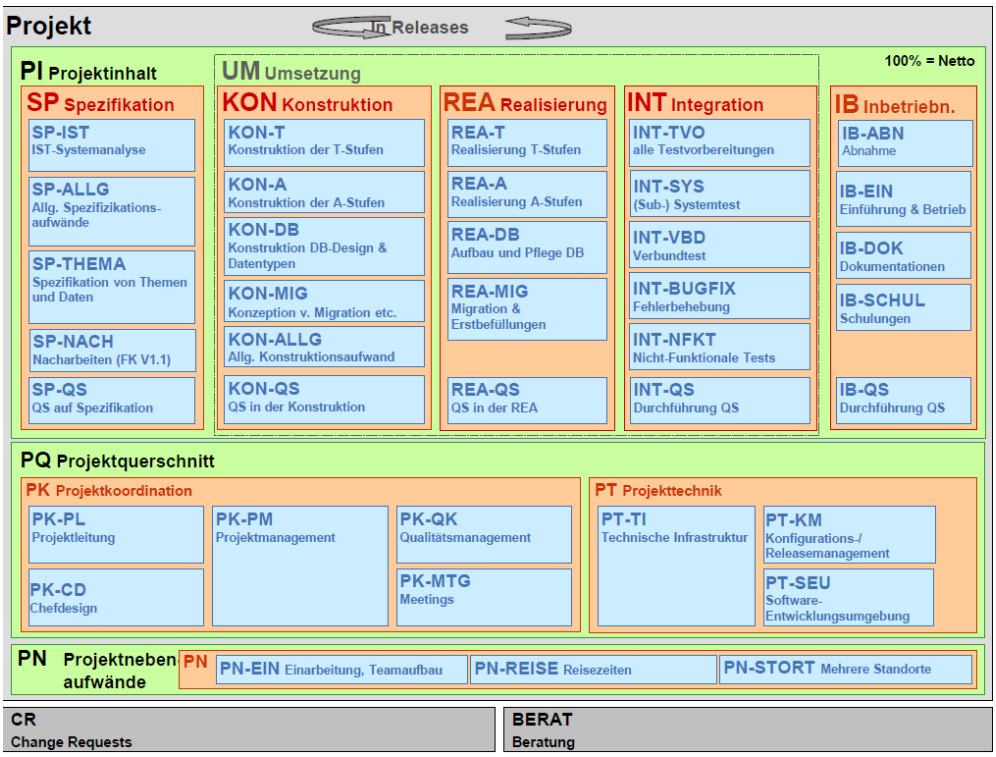
Interne Zinsfussmethode: $K = \sum_{t=0}^{n}\left[(E_t - A_t) * \frac{1}{(1+r)^t}\right] = 0$

K = Kapitalwert
Et = Einzahlungen am Ende der Periode t (Nutzen)
At = Auszahlungen am Ende der Periode t (Kosten)
r = Zinsfusssatz
t = Periode (t = 0, 1, 2 ..., n)
n = Nutzungsdauer des Investitionsobjektes

Bottom-Up: Aufwände werden für einzelne Module geschätzt und dann summiert. (Expertenschätzung, Delphi-Methode, Schätzklausur)

Top-Down: Gesamthafte Schätzung des Projektaufwandes mit Hilfe von mathematischen Algorithmen auf Basis der funktionalen Anforderungen. (COCOCMO, Function Points, Use Case Points, Analogiemethode, Multiplikatormethoden, Prozentsatzmethoden)

1BT = 8Bh
1BW = 40Bh = 5BT
1BM = 160Bh = 20 BT
1BJ = 1600Bh = 10BM



Ebene 0,
Für Kennzahlen und Auswertungen

Ebene 1

Ebene 2
Jedes Projekt schätzt und erfasst seine Aufwände auf einer dieser Ebenen. Ab einer Projektgröße von 15 BM ist die Ebene 2 verbindlich. Darunter kann beliebig detailliert werden.

→ **Die Ebene 1 & 2 definiert die Aufgabenkategorie**

Mobile Lösungen im Enterprise Einsatz

Comsumerisation of IT: Zuerst war IT in Unternehmen und wurde dann für die breite Masse zugänglich => Jetzt ist Technologie erst für private Nutzer da, bevor sie in Unternehmen gebrauch finden (Handy, Tablet, Neues Betriebssystem, …)

Mobile Clients: Unterscheiden sich in OS, Display, Akkulaufzeit, Prozessor, Ram, Speicher, Schnittstellen, Tastatur, Sturzschutz, IP-Schutzklasse, Art der Datenerfassung, Maße und Gewicht, Betriebstemperatur, Preis, …

Einsatz: Datenerfassung und Datendarstellung

Optimierung, Automatisierung, Beschleunigung, Qualität

Kosten und Bedienbarkeit steigen von: GUI (manuelle Eingabe) → Barcodes (Optische Erfassung) → RFID (Elektromagnetische Erfassung)

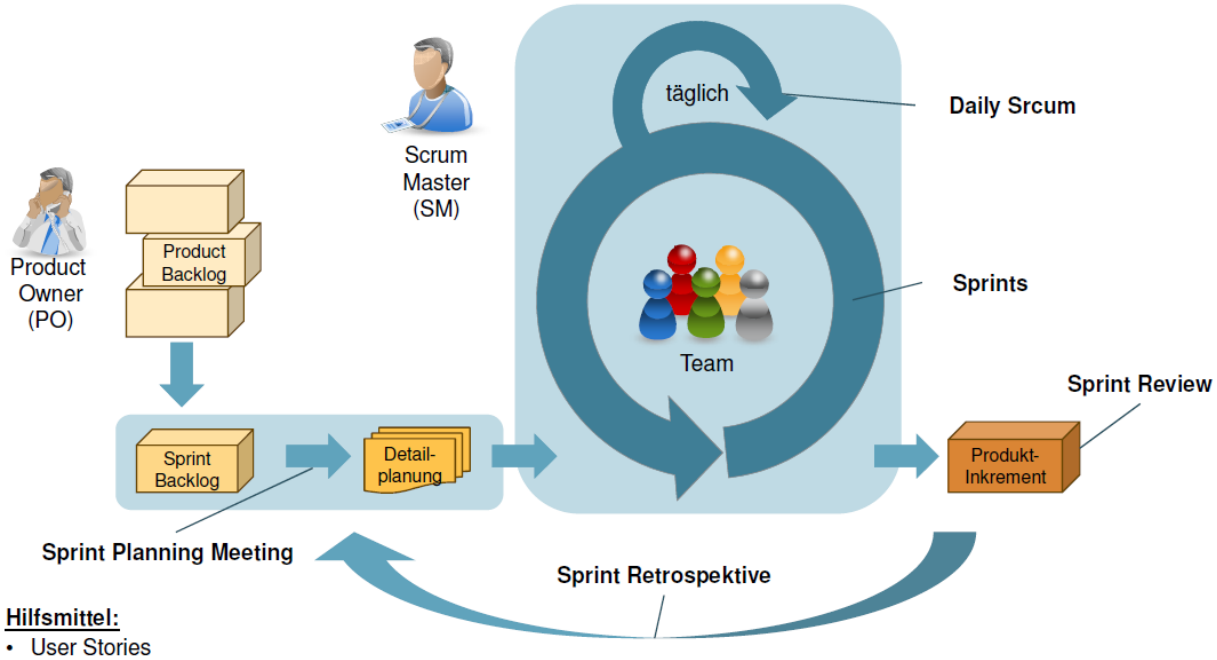
Prototyping: Schnell, einfach, günstig, leicht änderbar, kein Source-Code (Prototyp wird weggeworfen!!!)

Tools: Fluid UI, Balsamiq Mockups, Axure RP, Stift und Papier

SCRUM

Allgemeine Motivation: Beschleunigte Markteinführung, Verbesserter Umgang mit Änderungen, Umfang, Anforderungen, Prioritäten, Stärkere Einbindung der Kunden, Frühe Risikominimierung, Verbesserte Sichtbarkeit des Projekts, Kunden haben eine agile Unternehmenskultur

Klassisch	Iterativ	Agil
Team durch Projektleiter geführt	Team durch Projektleiter geführt	Transparenz und Kommunikation
Lange Entwicklungszyklen	Iterativ inkrementelles Vorgehen	Iterativ – Inkrementell (Sprints)
Wasserfall-ähnliche Konkretisierung über alle Bereiche	Priorisierung und Entwicklung von Anforderungen	Priorisierung und Entwicklung von Anforderungen
Gesteuert durch einen Plan	Gesteuert durch einen Plan	Anpassungsfähige Planung
Formaler CR Prozess und Nachvollziehbarkeit	Häufiges Kunden-Feedback	Aktive Kunden
Dokumentations- und Qualitätskontrollen von Zwischenergebnissen	Dokumentations- und Qualitätskontrollen von Zwischenergebnissen	Selbstständiges Team
Schwergewichtiger Prozess	Schwergewichtiger Prozess	Leichtgewichtiger Prozess



Hilfsmittel:

- User Stories
- Impediment List
- Burndown Charts
- Definition of Done

Entwicklungs-Team: 5-9 Vollzeitmitglieder, Funktionsübergreifend: QS; Programmierer, UI-Designer, …

Product Owner: erfasst Bedürfnisse der Kunden und Stakeholder, bestimmt Auslieferungsdatum und Inhalt, ist verantwortlich für Produkt und Projekterfolg, definiert und priorisiert Features abhängig vom Geschäftswert, akzeptiert oder weist Arbeitsergebnisse zurück

Scrum Master: ist verantwortlich für die Einhaltung der Scrum-Werte und Techniken, hilft beim Beseitigen der Hindernisse, schützt das Team vor äußeren Störungen, ABER: darf nicht inhaltlich arbeiten, ist nicht Teil des (Umsetzungs-)Teams, hat keine Weisungsbefugnis gegenüber dem Team

Product-Backlog: Enthält die zukünftigen Features in Form von User stories, wird vom Product Owner verwaltet, Features sind vom Product Owner priorisiert (wichtig […] unwichtig)

Definition of done: Definiert was mindestens erfüllt sein muss, damit etwas abgenommen wird.

Sprint Backlog: Wird vom Team verwaltet, enthält die User Stories welche vom Team für den aktuellen Sprint geplant sind. Jeder Task enthält seinen Aufwand.

Burndown Chart: Visualisierung bereits geleisteter und noch verbleibender Arbeit, für jeden Sprint und ggf. Release

Impediment List: Liste aller Hindernisse, wird vom Team verwaltet, soll dem Team helfen Team-interne Hindernisse selbst zu beheben und Team-externe Hindernisse zu kommunizieren.