Maintance

Phase VI

Kaizen

- „Veränderung, Wandel“, zen „zum Besseren“; „Veränderung zum Besseren“
- Lernen aus Fehlern ist das Stichwort

Definitionen

- **Maintenance**: Aufgabe zur Sicherung des aktuellen Zustand
- **Maintainability**: Bewertung der Einfachheit/Komplexität von Maintenance
- Allgemein: Wenig komponentenspezfische Themen → daher kurzer Überblick über die Themen für einen Software Architekten

Typen von Software Maintainance

- Corrective
- Perfective
- Adaptive
- Preventive maintenance

Corrective

- Analysieren und beheben von Fehlern in der Software
- Vermutlich in den meisten Projekten jener Typ von Maintainance, welcher **am meisten Zeit** in Anspruch nimmt
- Eigenschaften:
  - **Severity**: Wie schwer ist der Fehler für den laufenden Betrieb? (meist critical, high, medium, und low)
  - **Priority**: Reihung – Wann wird der Bug gefixed?

Perfective Maintenance

- Erforderlich bei neuen oder geänderten Anforderungen (Requirements)
- Wenn Maintainability (Quality-Attribute / nicht-funktionale Anforderung) hoch ist → Anpassungen gehen einfacher

Adaptive maintenance

- Anpassungen an neue Umgebungsbedinungen
  - Neues Betriebssystem
  - Neues Datenbanksystem
  - …
- Auch hier gilt wieder: Wenn Maintainability (Quality-Attribute / nicht-funktionale Anforderung) hoch ist → Anpassungen gehen einfacher

Preventive maintenance

- Probleme in der Zukunft vermeiden, indem die Qualität gesteigert wird
- E.g. durch Steigerung von Maintainability und Reliability
- Kann e.g. durch Refactoring von Komponenten / Architektur umgesetzt werden

Maintainability designen

- Complexity is your enemy …
- Ebenfalls ist es wahrscheinlicher, dass Komponenten mit höherer Komplexität mehr Fehler beinhalten …
  - Und natürlich: von anderen (neuen) Entwicklern schwerer zu verstehen sind (→ Zeit)
  - Und natürlich: schwerer zu testen sind (→ Testautomatisierung)
- Wie kann man die Komplexität senken?
  - Reduce size
  - Increase cohesion
  - Reduce coupling

Reduce size

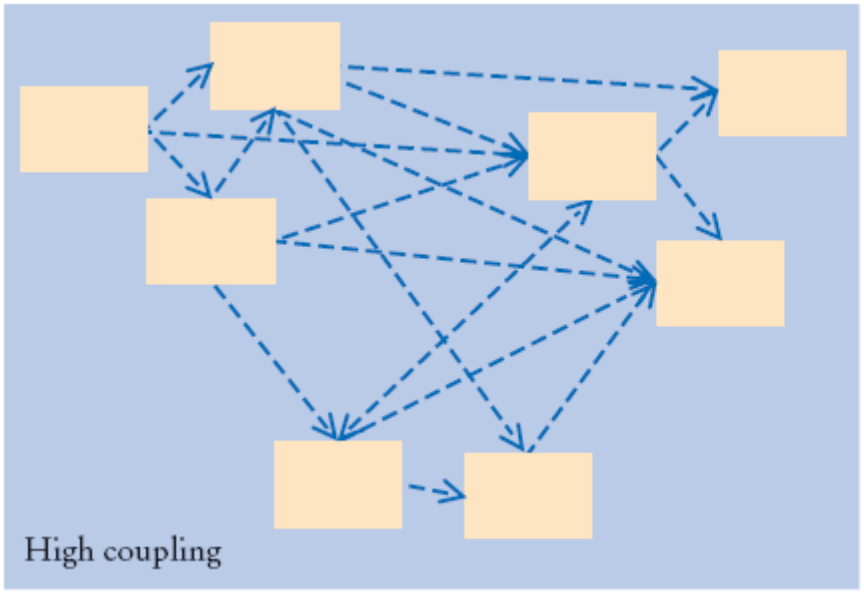
- Große Module (=viele Aufgaben) sind schwerer zu beherrschen als kleine
- Durch Refactoring kann man große Module in kleine Module unterteilen → Stichwort: kontinuierliche Verbesserung

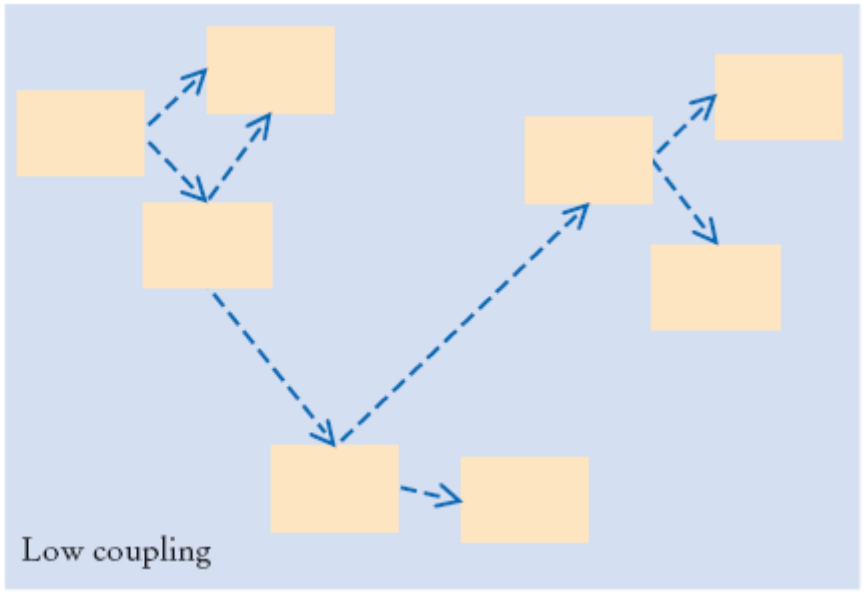
Increasing cohesion

- Die Aufgaben in einer Komponente sollten eng miteinander verknüpft sein → Aufgaben die miteinander nichts zu tun haben, sollten auch nicht in einer Komponente sein
- Es gibt viele Arten von Kohäsion (siehe https://de.wikipedia.org/wiki/Koh%C3%A4sion_(Informatik))
  - Logische Kohäsion liegt dann vor, wenn die – an sich unterschiedlichen – Teile eines Moduls logisch durch einen Oberbegriff zusammengefasst werden können (Beispiel: Eingaberoutinen für Maus, Tastatur etc.).
  - Funktionale Kohäsion liegt dann vor, wenn die Teile eines Moduls alle zur Lösung einer einzelnen, wohldefinierten Aufgabe beisteuern.
  - ….

Reducing coupling

- Wie stark sind Module voneinander abhängig?
- Ziel: so wenig wie möglich abhängig → Änderungen leichter durchzuführen





Weitere Themen / Stichworte

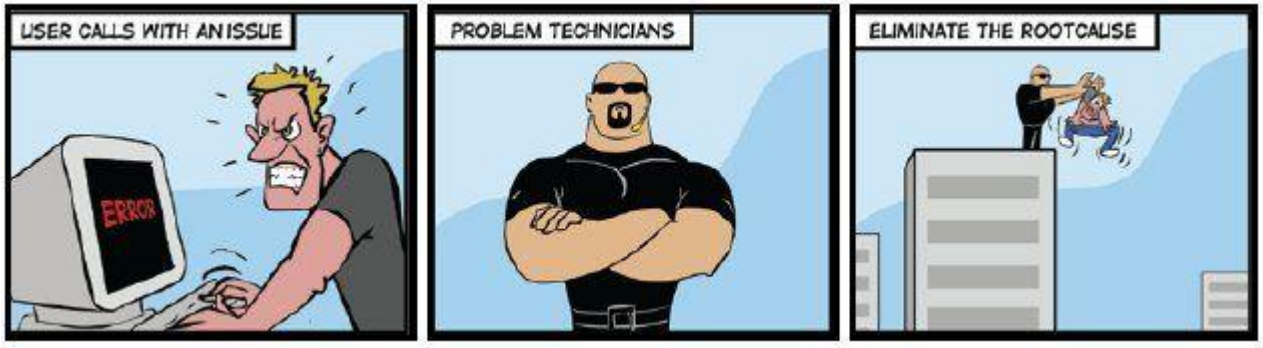
- Themen
  - Monitoring
  - Incident management
  - Problem management
  - Change management
- Themen auch in Information Technology Infrastructure Library (ITIL) → große Ähnlichkeiten

Incident management

- „Unter einem Incident versteht man die Störung des normalen Service, der den Benutzer oder das Business beeinträchtigt. Ziel von Incident-Management ist, den IT-Service so schnell wie möglich wieder in den **ursprünglichen Zustand zu versetzen - mit Workarounds oder Lösungen**, die sicherstellen, dass der Geschäftsbetrieb nicht beeinträchtigt wird.“
- https://www.manageengine.de/produkte-loesungen/it-helpdesk/servicedesk-plus/infomaterial/itil-handbuch.html

Problem management

- „Das Ziel des Problem-Managements ist es, die **Urasache des Incidents zu finden** und den Einfluss auf den Geschäftsbetrieb zu reduzieren. Problem-Management ist ein proaktiver Ansatz, der das Wiederauftreten der Incidents verhindern soll.“



Change management

- „Der Change-Management-Prozess hilft Ihnen, die Veränderungen (Changes) mit minimalen Beeinträchtigungen und Risiken zu koordinieren.“

Fazit

- Nachdem die Software beim Kunden ist, ist die Arbeit noch nicht getan
- Auch nicht, wenn er keine neuen Anforderungen mehr hat
- Die Ständige Verbesserung (Kaizen) sollte an oberster Stelle stehen