
Author

# Title

**Sub/Work Title**

University of Applied Sciences: Vorarlberg

Department of Computer Science
Professor / Supervisor

Dornbirn, 2014

# Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

# Inhaltsverzeichnis

# Listings

# Abbildungsverzeichnis

# Todo list

# 1 Einführung

Das Hochhalten von Softwarequalität ist eine sehr anspruchsvolle Aufgabe, welche eine gewisse Planung vorraussetzt. Gerade bei Projekten mit hohen Budgets und materiellen Einsätzen werden hohe Anforderungen an das Qualitätsmanagement gesetzt.

Einige bekannte Fehlschläge in der Softwareentwicklung hätten wahrscheinlich mit besseren Qualitätssicherungsmaßnahmen verhindert werden können:

- Pioneer 4 verfehlte den Mond
- unnötige Mahnungen durch die französische Finanzverwaltung
- das Herausgeben von faulen Krediten, was schlussendlich zum Bankencrash geführt hat
- Verlust einer Segelyacht im Pazifik

## 1.1 Verifikation und Validierung

Die Begriffe sollten klar getrennt werden, da sie nicht synonym verwendet werden können, und bei Softwareprozessen und Softwarequalität eine gewichtige Rolle spielen.

Die Verifikation stellt fest ob die Software mit der vorhandenen Spezifikation übereinstimmt, wohingegen die Validierung bestimmt ob die Software für den Kunden auch wirklich nützlich ist.

## 1.2 Technical Debt

Technical Debts sind ein wichtiges Thema in der Qualitätssicherung. Der Begriff wurde aus dem Finanzwesen übernommen (Debt = Schulden). In der Softwareentwicklungen ist damit gemeint, dass ungeschickte Lösungen irgendwann gefixt werden müssen.

Diese technische Schulden kann man bewusst eingehen, um Termine zu halten. Allerdings muss man immer im Hinterkopf behalten, dass man diese Schulden zu einem späteren Zeitpunkt auch wieder bezahlen muss.

Einer der größten Unterschiede zwischen klassischen und agilen Projektmanagementmethoden ist der Umgang mit diesen technischen Schulden. Bei agilen Methoden werden diese Schulden bewusst eingegangen, um ein schnelleres iteratives Vorgehen zu gewährleisten. Bei diesen Methodiken ist allerdings auch die Gefahr einer Überschuldung ungleich höher als bei den klassichen.

Allerdings werden in der Regel auch bei klassischen Methoden technische Schulden verursacht, nämlich in Form von Änderungen in den Anforderungen, auf welche in agilen Methoden besser reagiert werden kann.

# 2 Hausaufgabe Besprechnung

Es sollte klar definiert werden was in den Sonderfällen passiert bzw. was in diesen Fällen zurückgegeben wird (z.B. komplexe Zahlen da in Wurzel negative Zahl). Gleitkommzahlen sind "Teufelswerk"! Wenn man zwei gleiche Zahlen (die berechnet wurden aber unterschiedliche Rundungsfehler haben) voneinander voneinander subtrahiert bleibt nicht 0 übrig sondern der Rundungsfehler. Durch Rundungsfehler können somit 2 verschiedene Ergebnisse entstehen die eigentlich eines wären - darum könnte man anstatt der Mitternachtsformel eine andere Formel / einen anderen Ansatz verwenden (z.B. Newton-Näherungsverfahren).

> Äls Softwareentwickler ist es durchaus erlaubt sein Hirn einzuschalten! - Gleitkommzahlen sind Teufelswerk!"

# 3 Wichtige Begriffe

- Softwareprozesse (Abfolge von Tätigkeiten, durch die ein Software-Produkt entsteht)
- Vorgehensmodell (Vereinfachte Beschreibung eines Softwareprozesses)
- Methode (Strukturierter Ansatz für die Software-Entwicklung)

## 3.1 Software-Engineering

Software-Engineering ist eine technische Disziplin welche eine Lösung für den Anwender bieten will unter Einsatz von Theorien, Methoden und Werkzeugen und Berücksichtigung von Organisation, Management und Entwicklung.

## 3.2 Informatik

Grundlage der Wissenschaft für Software-Ingeneure (Theorie, Konzepte, ...).

## 3.3 ESSENCE Kernel Overview

In diesem Model gibt es verschiedene Zustände für die Anforderungen und jeder dieser Zustände hat selbst wieder Kriterien welche einem dabei helfen in welchem Zustein ein Projekt ist.

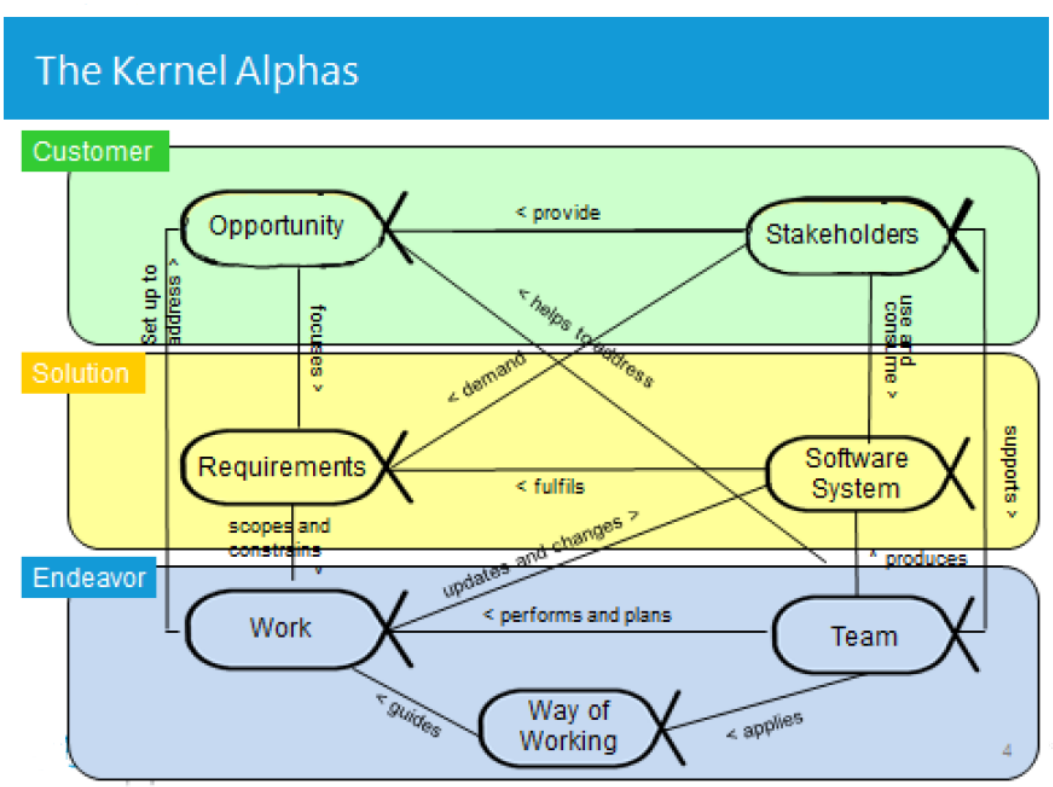


Abbildung 3.1: ESSENCE Kernel Overview

**Zustände:**

- Conceived
- Bounded
- Coherent
- Accepted
- Adressed
- Fulfilled

## 3.4 Qualität

Grad in dem die inhärenten Eigenschaften des Produkts Anforderungen erfüllen.

- **Explizite Anforderungen**
- **Implizite Anforderungen** - Anforderungen welche existieren aber den Stakeholdern nicht bewusst sind.
- **Nicht explizite Anforderungen** - Stakeholer wissen, dass sie diese Anforderungen gibt aber sie teilen diese nicht mit (sind quasis eh klar).
- **Objektiv** - Vollkommen klar, dass man etwas braucht.
- **Subjektiv** - Vermeintliche Anforderungen, welche nicht wirklich wichtig sind.
- alle betroffenen / interessierten Personen

### 3.4.1 Begriffsabgrenzung

**Technisches computer-basiertes System** System welches ausschließlich aus Soft- und Hardware-Komponenten besteht.

**Soziotechnisches System** System bestehend aus einem oder mehreren technischen Systemen, den Menschen die es bedienen, den notwendingen Arbeitsprozessen, organisatorischen Richtlinien, usw.

*Systeme:*

- haben systemspezifische Eigenschaften die nur dem System als Ganzem zugeordnet werden können
- sind häufig nicht deterministisch
- hängen von organisatorischen Zielen ab

## 3.5 Kritische Systeme

Relevante Systemeigenschaften für kritische Systeme sind:

- Reparierfähigkeit
- Wartbarkeit
- Überlebensfähigkeit
- Fehlertoleranz

**Kritisches System** Systeme, bei dessen Ausfall oder Fehlfunktion großen Schaden anrichten kann (wirtschaftliche Verluste, physische Schäden, Gefahr für Gesundheit und Leben von Menschen).

**Sicherheitskritisches System** Schäden an der Umwelt und/oder Gefahr für Gesundheit und Leben von Menschen (Bohrinsel im Golf von Mexiko).

**Aufgabenkritisches System** Aufgaben die ein System erledigen solle werden nicht durchgeführt (z.B. Bank).

**Geschäftskritisches System** Extrem hohe Kosten bzw. signifikante Gewinnausfälle können die Folge eines Systemausfalls sein.

# 4 Zusammenfassung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.