



Programmierskills

Inhaltsverzeichnis

1. Code – Konzeption

1. *Softwarearchitektur*
2. *KISS – Prinzip*
3. *Coupling – Cohesion*
4. *Code - Pasta*

2. Programmierung

1. *Pair Programming*
2. *Distributed Programming*
3. *Refactoring*
4. *Code - Review*

1. Code – Konzeption

1.1 *Software - Architektur*

- Grundlegendste Designentscheidung
- Struktur der Systemkomponenten
 - Beschreibung von deren Beziehungen
- Globale Eigenschaft des Gesamtsystems
- meist nur schwer veränderbar
 - Abänderbarkeit als Qualitätsmerkmal

1. Code – Konzeption

1.2 *KISS - Prinzip*

- Bevorzugen einfacher Problemlösungen
- Ähnlichkeit zu „Ockhams Rasiermesser“ (14. Jh.)
- Urheber Ingenieur Clarence Johnson
- Auslegungen:
 - „*Keep it simple, stupid*“
 - „*Keep it simple [and] stupid*“
 - „*Keep it simple and smart*“

1. Code - Konzeption

1.3 Cohsion - Coupling

- Maß zur Bewertung von Softwarequalität
- ***Coupling*** - Grad der Interdependenz zwischen zwei Softwaremodulen
- geringes ***Coupling*** Zeichen für gute Struktur
- ***Cohesion*** – Grad der Stärke der Beziehung verschiedener Elemente eines Moduls
- starke ***Cohesion*** und geringes ***Coupling*** erstrebenswert

1. Code – Konzeption

1.4 Code Pasta

- **Spaghetti-Code:** in sich verworrener Code
 - schlechte Les- und Erweiterbarkeit
- **Lasagna-Code:** Code mit zu vielen „Schichten“
 - Ineffektiv und schlecht Lesbar
- **Ravioli-Code:** viele eigenständige Elemente
 - gute Wiederverwertbarkeit und Verständlichkeit
 - gute Wartbarkeit (geringes Coupling)
- **Macaroni-Code:** mehrere Sprachen in einer Datei
 - gute Sprachkenntnisse nötig
 - Sehr effektiv und übersichtlich
 - Code Komponenten-weise bündelbar

From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.

END

I have no memory

of this code

quickmeme.com

2. Programmierung

2.1 *Pair - Programming*

- 2 Programmierer an einem PC
- einer schreibt Code, der andere kontrolliert
 - Diskussion über Probleme
- Vorteile:
 - Besserer Code
 - Weniger Fehler
 - Wissensvermittlung
 - Teambildung
- Nachteile:
 - schwierige Teamfindung
 - Urheberrechts- und Haftungsunklarheit
 - langsamer

2. Programmierung

2.2 *Distributed Programming*

- Aufteilung der Aufgaben auf mehrere Programmierer
- jeder erstellt Code und kontrolliert ihn
- Vorteile:
 - effizienteres Arbeiten
 - modularer Code
 - große Flexibilität
- Nachteile:
 - geringer Lerneffekt
 - wenig Teamwork

2. Programmierung

2.3 Refactoring

- Verfahren zur Strukturverbesserung von Code
 - Erhaltung des Programmverhaltens!
- Verbesserung von:
 - Lesbarkeit
 - Wartbarkeit
 - Erweiterbarkeit
- Teil der „agilen“ Softwareentwicklung

2. Programmierung

2.4 Code - Review

- manuelle Durchsicht des Quellcodes
 - einer der mehrere Tester
 - Diskussion über Ergebnisse
 - Fehlerbehebungen durch Autor
- gute Weiterbildungsmöglichkeit
- „statische“ Testmethode (keine Ausführung)

Quellen

- <https://de.wikipedia.org/wiki/Softwarearchitektur> (09.07.16)
- <https://de.wikipedia.org/wiki/KISS-Prinzip> (09.07.16)
- <https://de.wikipedia.org/wiki/Refactoring> (09.07.16)
- <https://de.wikipedia.org/wiki/Paarprogrammierung> (09.07.16)
- https://en.wikipedia.org/wiki/Coupling_%28computer_programming%29 (09.07.16)
- https://en.wikipedia.org/wiki/Cohesion_%28computer_science%29 (09.07.16)
- <http://www.docsity.com/en/news/programming-2/programming-pasta-spaghetti-lasagna-ravioli-macaroni-code/> (09.07.16)
- [https://de.wikipedia.org/wiki/Review_\(Softwaretest\)](https://de.wikipedia.org/wiki/Review_(Softwaretest)) (24.07.16)