University of Basel

Software Engineering

Marcel Lüthi, Universität Basel

Validierung und Verifikation

Validierung und Verifikation

Validierung

- Bauen wir das richtige Produkt?
- Können Nutzer die festgelegten Ziele erreichen?

Verifikation

- Bauen wir das Produkt richtig?
- Erfüllt Produkt spezifizierte Eigenschaften?

Beispiel: Zulassung Medizinprodukt

- Nachweis des medizinischen Nutzens (Validierung)
- Nachweis, dass Funktion erfüllt wird (Verifikation)

Validierung vs Verifikation

Validierung

- Bauen wir das richtige Produkt?
- Können Nutzer die festgelegten Ziele erreichen?

Quelle: https://www.trendhunter.com/trends/umbrella-shoes

Verifikation

- Bauen wir das Produkt richtig?
- Erfüllt Produkt spezifizierte Eigenschaften?

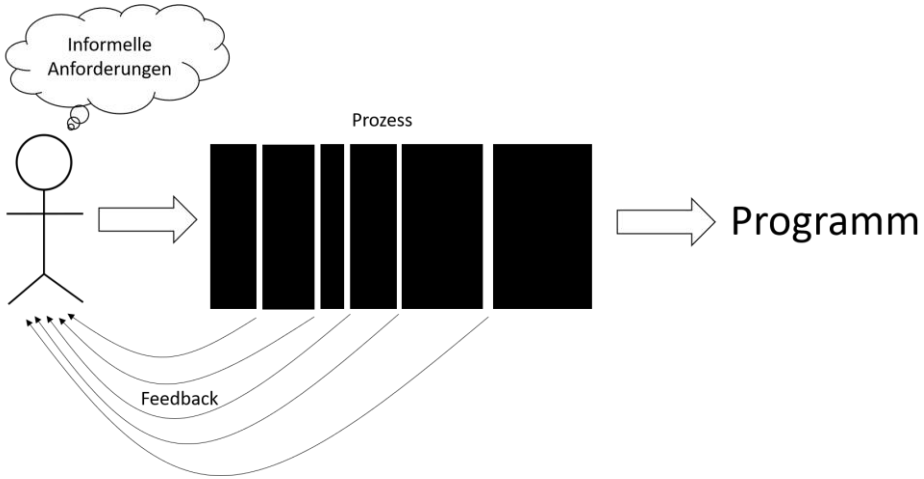
https://www.boredpanda.com/useless-object-design-the-unusable-katerina-kamprani/l&utm_campaign=organic

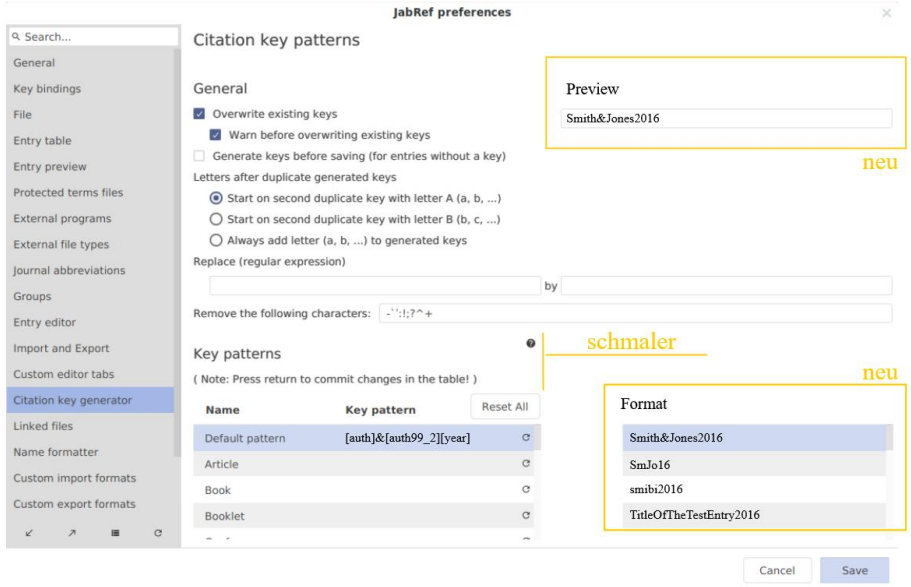
Validierung

Insgesamt schwierig: Kaum formale Dokumente / Spezifikationen auf denen die Anforderungen aufbauen

Ansätze:

- Frühe Prototypen / Modelle
- Inkrementelles Entwickeln
- Reviews
- Simulationen





Quelle: André Nuber, David Tyndall , Software Engineering course 2022

Warum braucht es Verifikation?

Fehler lassen sich nie ganz verhindern

Mögliche Fehlerquellen

- Unklarheiten / Mehrdeutigkeiten
- Falsche Annahmen
- Falsche Schlussfolgerungen / Fehler
- Fehler in Systemumgebung

Ansätze

Analyse (statisch)

- Code reviews
- Automatisierte Codeanalyse
- Formale Korrektheitsbeweise

Experimentieren (dynamisch)

- Verhaltens des Programms testen
  - Manuelle Ausführung
  - Automatisierte Tests

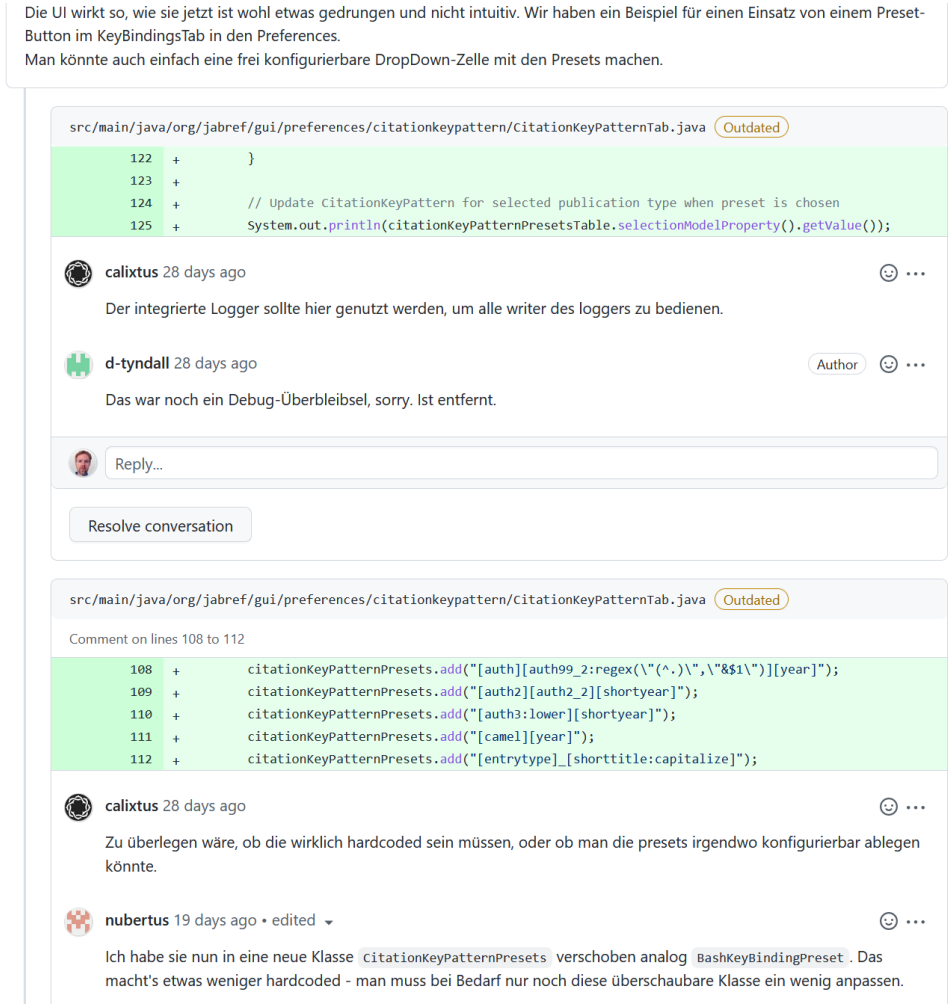
Code Review

Durchsicht des Programms durch andere Person(en)

Ziele

- Missverständnisse und Fehler entdecken
- Codequalität verbessern
- Zusammenarbeit stärken
- Wissen verbreiten

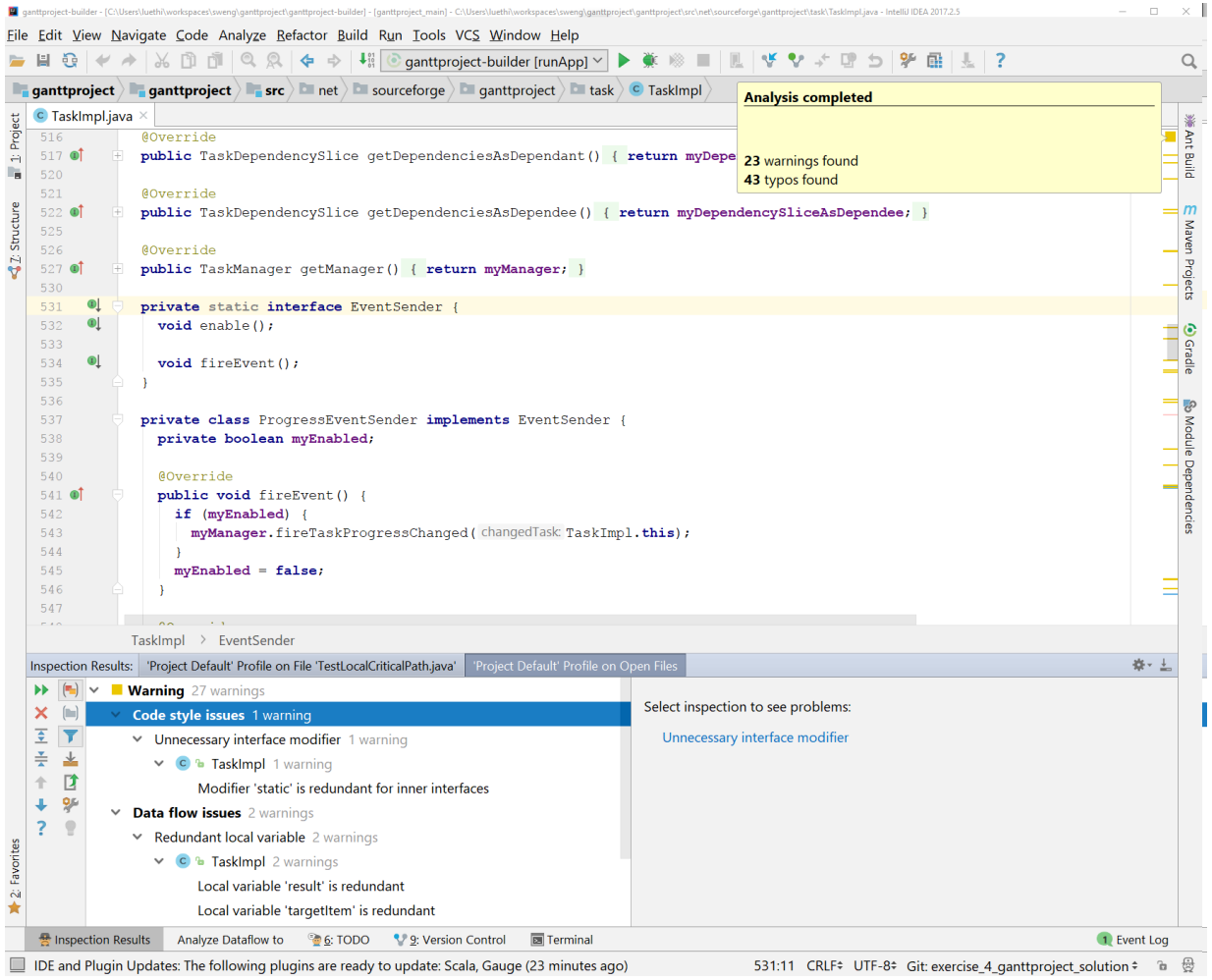
Oft unterstützt durch Online-Tools (hier Github)



(Statische) Codeanalyse

- Durchgehen vom Code. Prüfen auf typische Probleme.
  - Uninitialisierte Variablen
  - Index out of bounds
  - Möglicher Zugriff auf Null-Werte
  - Unbenutzte Methoden
  - ...

- Oft durch Entwicklungstools unterstützt.



Korrektheitsbeweise

```
{true}
begin
      read(a); read(b);
      x = a + b
      write(x);
end
{output = input1 + input2}
```

Ziel: Beweisen, dass Programm Spezifikationen erfüllt.

- Spezifikationen müssen formal definiert sein.
- Braucht "automatische Theorembeweiser" für nicht-triviale Programme

Testen

Verhalten eines Programms durch Stichproben von Eingaben überprüfen.

Ziel: Beispiele von inkorrektem Verhalten finden

Quelle: https://promwad.com/news/testlink