



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?



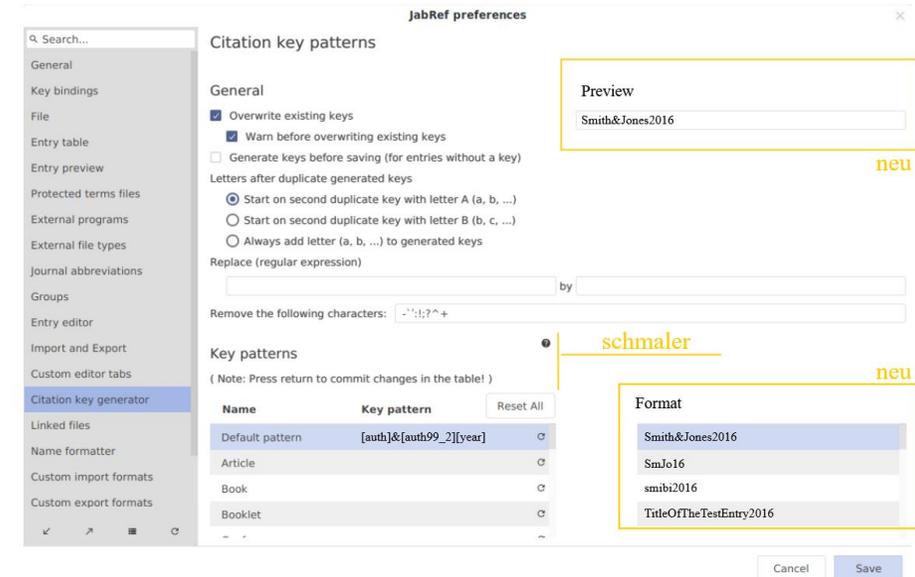
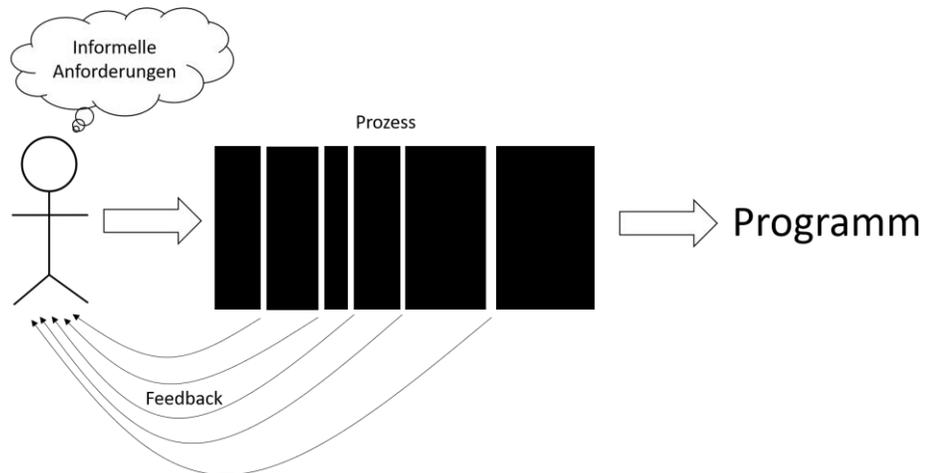
https://www.boredpanda.com/useless-object-design-the-unusable-katerina-kamprani/l&utm_campaign=organic

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)

Die UI wirkt so, wie sie jetzt ist wohl etwas gedrungen und nicht intuitiv. Wir haben ein Beispiel für einen Einsatz von einem Preset-Button im KeyBindingsTab in den Preferences.
Man könnte auch einfach eine frei konfigurierbare DropDown-Zelle mit den Presets machen.

```
src/main/java/org/jabref/gui/preferences/citationkeypattern/CitationKeyPatternTab.java Outdated  
122 +     }  
123 +  
124 +     // Update citationKeyPattern for selected publication type when preset is chosen  
125 +     System.out.println(citationKeyPatternPresetsTable.selectionModelProperty().getValue());
```

 calixtus 28 days ago 😊 ...

Der integrierte Logger sollte hier genutzt werden, um alle writer des loggers zu bedienen.

 d-tyndall 28 days ago Author 😊 ...

Das war noch ein Debug-Überbleibsel, sorry. Ist entfernt.

 Reply...

Resolve conversation

```
src/main/java/org/jabref/gui/preferences/citationkeypattern/CitationKeyPatternTab.java Outdated
```

Comment on lines 108 to 112

```
108 +     citationKeyPatternPresets.add("[auth][auth99_2:regex(\"^(^)\",\"&$1\")][year]");  
109 +     citationKeyPatternPresets.add("[auth2][auth2_2][shortyear]");  
110 +     citationKeyPatternPresets.add("[auth3:lower][shortyear]");  
111 +     citationKeyPatternPresets.add("[camel][year]");  
112 +     citationKeyPatternPresets.add("[entrytype]_[shorttitle:capitalize]");
```

 calixtus 28 days ago 😊 ...

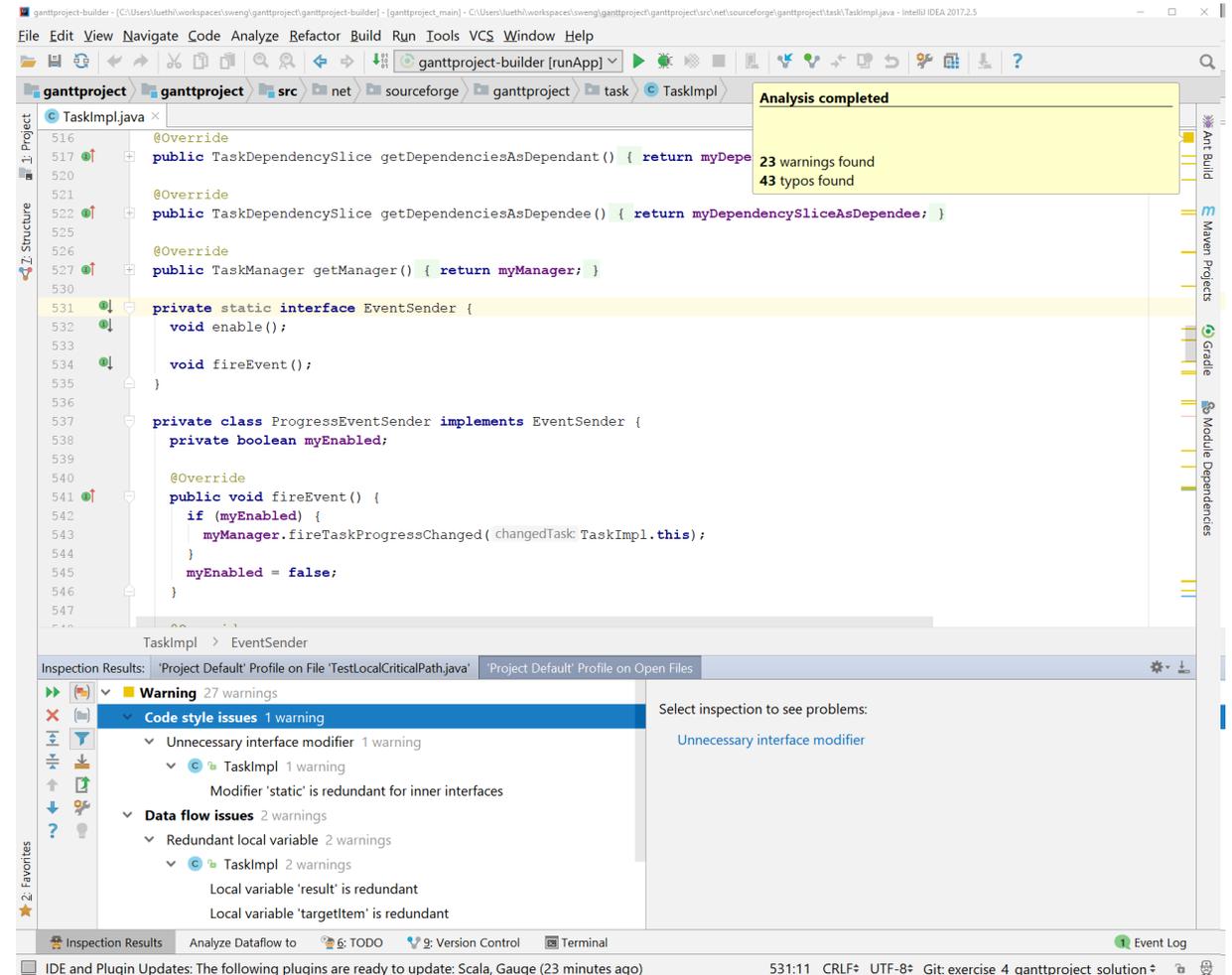
Zu überlegen wäre, ob die wirklich hardcoded sein müssen, oder ob man die presets irgendwo konfigurierbar ablegen könnte.

 nubertus 19 days ago • edited 😊 ...

Ich habe sie nun in eine neue Klasse `CitationKeyPatternPresets` verschoben analog `BashKeyBindingPreset`. Das macht's etwas weniger hardcoded - man muss bei Bedarf nur noch diese überschaubare Klasse ein wenig anpassen.

(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

```
100 little bugs in the code, 100 bugs in the code,  
fix one bug, compile it again, 101 little bugs in the code.  
101 little bugs in the code... Repeat until BUGS = 0
```

