

# Refaktorisierung einer Architekturanalyse für Vertraulichkeit

Praktikum Ingenieurgemäße Softwareentwicklung

Betreuer: Frederik Reiche

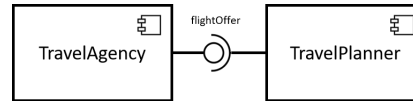
Alina Valta | 10. März 2022



# Motivation

Komponenten basierte Softwareentwicklung:

- Wiederverwendbare Komponenten
- Komposition von Komponenten



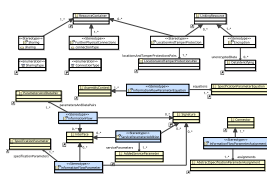
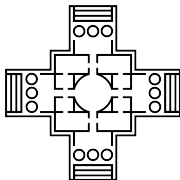
Vertraulichkeitsanalyse (Kramer, Hecker, Greiner u. a. 2017)

- Vertrauliche Daten überschreiten Komponenten-Grenzen
- Probleme auf Architekturebene erkennen

⇒ kompositorische Analyse notwendig

# Vertraulichkeitsanalyse

## Architekturebene



```
dataSet(2).  
parametersAndDataPair(8).  
parameterSources(8,[return]).  
dataTargets(8,[5,6,7,4]).
```

```
isInSecureWithRespectTo(guest)  
+- accessibleParameters(guest,return(getId))  
| +- linksDataAccessibleBy(guest,wireless)  
| | +- linkAccessibleBy(guest,wireless)  
| | | +- linkLocation(wireless) | | | '-  
locationsAccessibleBy(guest)
```

### Palladio Component Model

### Confidentiality Model

### Prolog Prädikate

### Analyse Ergebnis

→  
Confidentiality4CBSE<sup>1</sup>

→  
PCM2Prolog<sup>2</sup>

→  
Haskalladio

<sup>1</sup><https://github.com/KASTEL-SCBS/Confidentiality4CBSE>

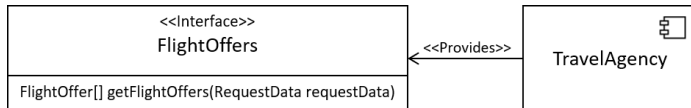
<sup>2</sup><https://github.com/KASTEL-SCBS/PCM2Prolog>

### DataSet

- Menge an Ein- und Ausgabedaten von Komponenten
- Trennung nach Benutzergruppen, Zweck der Informationen, ...

### InformationFlow

- Ordnet Daten DataSets zu
- Art der Daten:
  - Parameter
  - Rückgabewert
  - Aufruf der Funktion
  - Größe von Parametern



# Vertraulichkeitsmodellierung

## Maßnahmen, Angreifer und Ressourcen

### Location

- Geographische Orte oder Sicherheitslevel

### TamperProtection

- Maßnahmen gegen Manipulation

### Angreifer und Benutzer des Systems

- Welche DataSets dürfen bekannt sein?
- Welche TamperProtections kann/will der Angreifer umgehen?
- Zu welchen Locations hat der Angreifer Zugriff?

### Ressourcen

- Weitere Verbindungen, gleichzeitig laufende Programme
- Verschlüsselung

Vertraulichkeitsanalyse

○○●

Entfernen Profil-Mechanismus

○○

Information Modellierung

○○

PCM2Prolog

○○

Evaluierung

○○

Fazit

○

Literatur

# Bisheriges Modell

## Profil-Mechanismus

### Confidentiality Modell:

- Definiert Klassen zum Modellieren von DataSet, Maßnahmen, Angreifer, ...

### Confidentiality Profil:

- Verbindet PCM Elemente und Confidentiality Modell
- Zusammenfassung von mehreren Stereotypen
  - Stereotyp erweitert eine oder mehrere PCM Klassen
  - Stereotyp hat Referenzen zu Elementen aus dem Confidentiality Modell

⇒ Probleme mit Eclipse und unübersichtlich

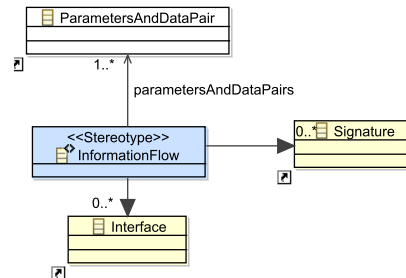


Abb: Beispiel Stereotyp

Confidentiality Modell:

- Definiert Klassen zum Modellieren von DataSet, Maßnahmen, Angreifer, ...

Confidentiality Profil:

- Verbindet PCM Elemente und Confidentiality Modell
- Zusammenfassung von mehreren Stereotypen
  - Stereotyp erweitert eine oder mehrere PCM Klassen
  - Stereotyp hat Referenzen zu Elementen aus dem Confidentiality Modell



**Lösung:** Stereotypen ersetzen

- Neue Modell Klassen
- Existierende Klassen erweitern
- Referenz zu PCM Elementen

⇒ Probleme mit Eclipse und unübersichtlich

# Entfernen Profil-Mechanismus

## Beispiel InformationFlow

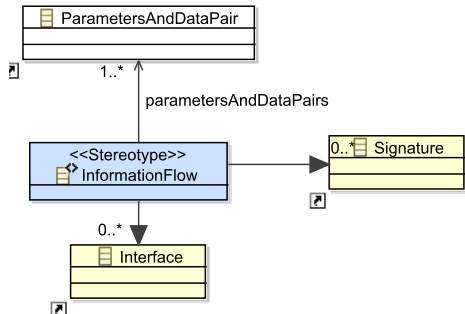


Abb: InformationFlow Stereotyp kann auf Signaturen und Interfaces angewandt werden

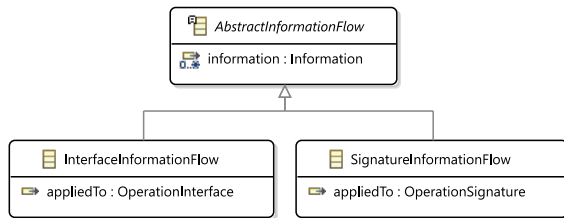


Abb: Modell nach Entfernen des Stereotyps  
Information Klasse ersetzt ParameterAndDataPair



Modellieren der Ein- und Ausgabedaten von Komponenten

- Zuordnung von DataSets und Daten einer Operation erfolgt über Strings
  - "requestData"
  - "\return"
  - "\call"
  - "\*"
  - "sizeof(\*)"
- Probleme:
  - implizite Referenzen
  - Syntax muss bekannt sein
  - Verwechslungsgefahr bei gleichnamigen Parametern

⇒ soll explizit modelliert werden

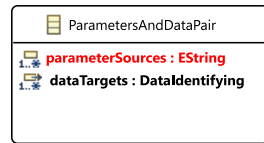


Abb: Zuordnung von Daten und DataSets über Strings im ursprünglichen Modell

# Informations-Modellierung

## String durch Referenz ersetzen

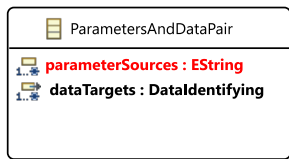


Abb: Zuordnung von Daten und DataSets über Strings im ursprünglichen Modell

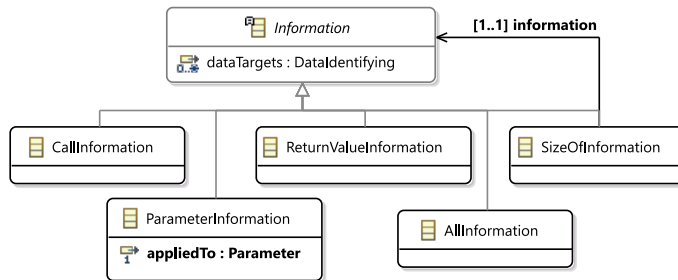


Abb: Modellierung der Information als eigene Klassen im neuen Modell

- Geschrieben in Xtend
- Reflective-API wird verwendet um Entitäten auf Prolog Prädikate abzubilden
- Filter bestimmt relevante Entitäten und Referenzen



```
parametersAndDataPair(10).
parameterSources(10,[return]).
dataTargets(10,[5]).

dataSet(5).
```

Abb: Beispiel Modellinstanz

Filter Entitäten: ParameterAndDataPair, DataSet

Filter Referenzen: id, dataTargets

Abb: Prolog Prädikate für diese  
Modell-Instanz

Änderungen des Modells sollen sich nicht auf den Prolog Code auswirken

- Dispatch-Methoden für Entitäten, die nicht automatisch generiert werden können

```
def dispatch String generateDeeplyCorrectly(EObject e) { return super.generateDeeply(e) }  
def dispatch String generateDeeplyCorrectly(AbstractResourceProtection rp) {...}
```

- Richtung der ursprüngliche Stereotypen Referenzen

- *vorher:* PCM  $\rightarrow$  Modell (über Stereotyp)

*jetzt:* Modell  $\rightarrow$  PCM

- Map für jeden früheren Stereotyp

- Key: PCM Element, Value: Set an Ids

- Beim Verarbeiten der Confidentiality Klassen wird die Map gefüllt

- Am Ende: erzeuge Prädikate aus den Map Elementen

Modellierung der Beispiel Projekte mit dem neuen Modell:

- Gleiche Ids verwenden

Automatischer Vergleich des Prolog Codes:

- Prolog Datei vorverarbeiten:

- Listen innerhalb von Prädikaten sortieren:

```
prädikatName(5, ["b","c","a"]). ⇒  
prädikatName(5, ["a","b","c"]).
```

- Zeilen der Datei sortieren
  - Leerzeilen entfernen

- Ausgabe mit diff vergleichen

Projekte `cloudscenario-minimized`<sup>1</sup> und `iflowexample`<sup>2</sup>:

- Gleicher Prolog Code konnte generiert werden

Grenzen der Refaktorisierung

- Kommentare
- Wiederverwendung von Hilfsklassen (z.B. `ParameterAndDataPairs`) nicht mehr möglich  
→ Prolog Code ändert sich

⇒ Bis auf die Wiederverwendung von Hilfsklassen können alle Modellinstanzen in das neue Modell übertragen werden, ohne dass der Prolog Code sich ändert.

---

<sup>1</sup><https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.cloudscenario-minimized>

<sup>2</sup><https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.iflowexample>

## Problem:

- Profil-Mechanismus + Eclipse
- String Referenzen

## Vorgehen:

- Neue Modell Klassen statt Stereotypen
- Explizite Referenzen statt Strings
- PCM2Prolog angepasst um Prolog Code nicht zu verändern





## Idee:

- Refaktorisierung des Modells
- Ersetzen des Profil-Mechanismus

## Ergebnis:

- Erfolgreiche Refaktorisierung zweier Beispiel Projekte
- Gleicher Prolog Code bis auf die Wiederverwendung von Hilfsklassen

# Referenzen

- 
Max E. Kramer, Martin Hecker, Simon Greiner u. a. *Model-Driven Specification and Analysis of Confidentiality in Component-Based Systems*. Techn. Ber. Karlsruhe: Karlsruhe Institute of Technology, Department of Informatics, 2017. DOI: 10.5445/IR/1000076957. URL: [http://dx.doi.org/10.5445/IR/1000076957%20\[Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen\]](http://dx.doi.org/10.5445/IR/1000076957%20[Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen]).
- 
Max E. Kramer, Martin Hecker und Frederik Reiche. *PCM2Prolog*. <https://github.com/KASTEL-SCBS/PCM2Prolog.git>.
- 
Max E. Kramer, Martin Hecker und Kateryna Yurchenko. *Examples4SCBS*. <https://github.com/KASTEL-SCBS/Examples4SCBS.git>.
- 
Max E. Kramer, Heiko Klare u. a. *Confidentiality4CBSE*. <https://github.com/KASTEL-SCBS/Confidentiality4CBSE.git>.