Karlsruher Institut für Technologie

Refaktorisierung einer Architekturanalyse für Vertraulichkeit

Praktikum Ingeneursgemäße Softwareentwicklung

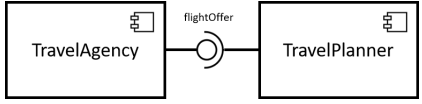
Betreuer: Frederik Reiche

Alina Valta | 10. März 2022

Motivation

Komponenten basierte Softwareentwicklung:

- Wiederverwendbare Komponenten
- Komposition von Komponenten



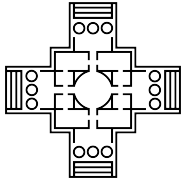
Vertraulichkeitsanalyse (Kramer, Hecker, Greiner u. a. 2017)

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

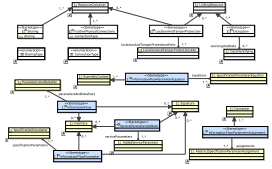
$\Rightarrow$ kompositorische Analyse notwendig

Vertraulichkeitsanalyse

Architekturebene



Palladio Component Model



Confidentiality Model

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

Prolog Prädikate

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

| | | +- linkLocation(wireless) | | | '-

locationsAccessibleBy(guest)
```

Analyse Ergebnis

$\longrightarrow$ Confidentiality4CBSE[1] $\longrightarrow$ PCM2Prolog[2] $\longrightarrow$ Haskalladio

[1] https://github.com/KASTEL-SCBS/Confidentiality4CBSE
[2] https://github.com/KASTEL-SCBS/PCM2Prolog

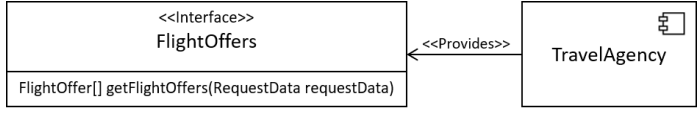
Vertraulichkeitsmodellierung
Datenfluss

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

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

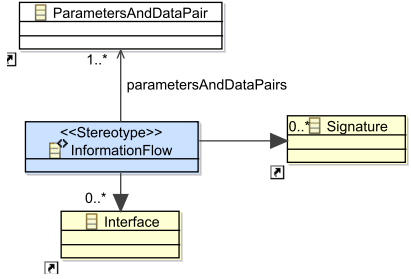


Abb: Beispiel Stereotyp

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

$\Rightarrow$ Probleme mit Eclipse und unübersichtlich

$\Rightarrow$

Lösung: Stereotypen ersetzen

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

Entfernen Profil-Mechanismus

Beispiel InformationFlow

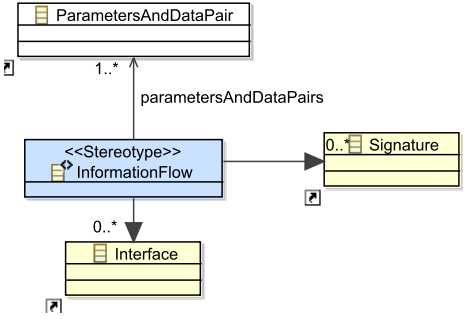


Abb: InformationFlow Stereotyp kann auf Signaturen und Interfaces angewandt werden

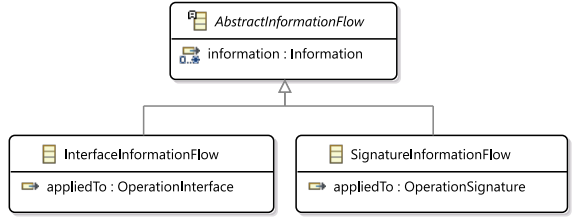


Abb: Modell nach Entfernen des Stereotyps Information Klasse ersetzt ParameterAndDataPair

Bisheriges Modell
Information Modellierung

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

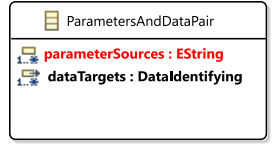


Abb: Zuordnung von Daten und DataSets über Strings im ursprünglichen Modell

Informations-Modellierung

String durch Referenz ersetzten

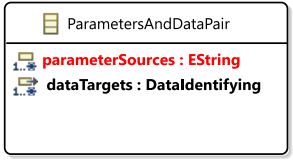


Abb: Zuordnung von Daten und DataSets über Strings im ursprünglichen Modell

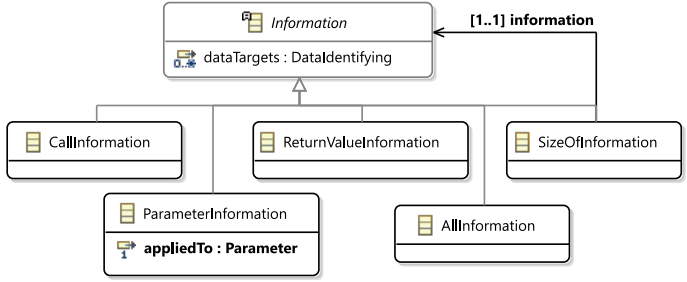


Abb: Modellierung der Information als eigene Klassen im neuen Modell

PCM2Prolog
Modellinstanz zu Prolog Code

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

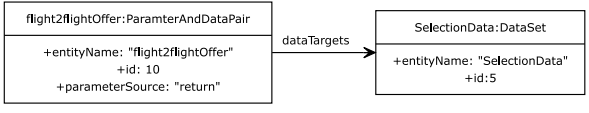


Abb: Beispiel Modellinstanz
Filter Entitäten: ParameterAndDataPair, DataSet
Filter Referenzen: id, dataTargets

→

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

dataSet(5).
```

Abb: Prolog Prädikate für diese Modell-Instanz

PCM2Prolog
Anpassungen

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 → Modell (über Stereotyp)
    jetzt: Modell → 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

Evaluierung
Automatisch Überprüfung der Ergebnisse

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

Evaluierung
Ergebnisse

Projekte `cloudscenario-minimized`[1] und `iflowexample`[2]:

- Gleicher Prolog Code konnte generiert werden

Grenzen der Refaktorisierung

- Kommentare
- Wiederverwendung von Hilfsklassen (z.B. ParameterAndDataPairs) nicht mehr möglich
  $\rightarrow$ PrologCode ändert sich

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

[1] https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.cloudscenario-minimized
[2] https://github.com/KASTEL-SCBS/Examples4SCBS/tree/master/bundles/edu.kit.kastel.scbs.iflowexample

Fazit

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

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].


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.