Sirius-Editor für erweiterte Datenflussdiagramme

Katrin Bott

14. September 2020

Institut für Programmstrukturen und Datenorganisation (IPD)
Praktikum: „Werkzeuge für agile Modellierung “
Betreuender Mitarbeiter: M.Sc. Stephan Seifermann

1 Einführung

Datenflussmodelle beschreiben Systeme aus einer funktionalen Sicht mittels ausgetauschter Daten. Sie finden unter anderem im Requirements Engineering als auch in der Sicherheitsanalyse ihrer Anwendung. In einem klassischen Datenflussdiagrammmodell wird keine Unterscheidung getroffen *welche Eigenschaften* die transportierten Daten haben.

In dem kleinen Anwendungsbeispiel in 1 lässt sich gut veranschaulichen, dass sich die Daten insbesondere im Bezug auf ihre Vertraulichkeit deutlich unterscheiden. Die Kredikartendaten eines Buchenden sind deutlich schützenswerter als die öffentlich zugänglichen Daten über einen Flug. Mit einem erweiterten Datenflussmodell lassen sich Charakteristiken wie hier die Zugriffsrechte repräsentieren und erlauben somit mehr Aussagen über die Sichherheitseigenschaften geplanter Systeme zu treffen.
Daraus ergibt sich die Aufgabenstellung einen Sirius-Editor für erweiterte Datenflussdiagramme zu entwickeln. Im letzten Semester wurde ein graphischer Sirius-Editor für

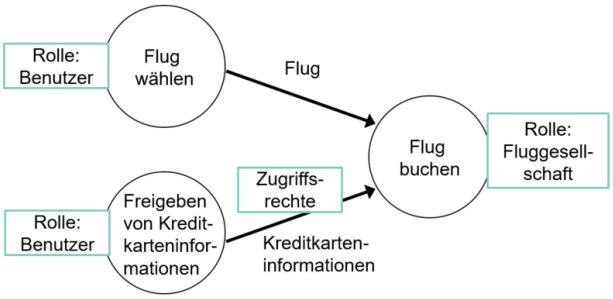

Abbildung 1: Datenflussdiagramm für eine Flugbuchung

Datenflussdiagramme entwickelt, der die Verfeinerung von Prozessen und Datenflüssen unterstützt. Dieser Editor sollte im Rahmen des Praktikums um verschiedene neue Elemente wie

- Charakteristiken
- Pins zum Datenaustausch
- Verhaltensbeschreibung von Knoten

erweitert werden und die Verfeinerung an das erweiterte Modell angepasst werden.

2 Grundlagen

2.1 Meta-Modelle

Im Folgenden werden die zugrundeliegenden Meta-Modelle und das Framework Eclipse-Sirius erläutert.

2.1.1 DataFlowDiagram

Das ursprüngliche Meta-Modell und der Editor befinden sich hier [2]. Dabei werden die vier grundlegenden Elemente eines klassischen Datendiagramms definiert

- **Externer Aktor**: Einführen von Daten in das System
- **Prozess**: Transformation eingehender und ausgehender Datenflüsse
- **Speicher**: Lesen-/Schreiben von Daten
- **Datenflüsse**: Transport von Daten von einer Quelle zu einem Ziel

und die Hierarchisierung von Prozessen und Datenflüssen definiert.

2.1.2 DataFlowDiagramCharacterized

Das erweiterte Modell ist hier [1]. Knoten und Datenflüsse können näher durch eine Menge von Charakteristiken beschrieben werden. Die Datenflüsse verlaufen von Output zu Input-Pins. Für die Knoten können Verhalten definiert werden. In einer Verhaltensdefinition befindet sich die Menge der In- und Output-Pins und die Definition von Zuweisungen. Zuweisungen bestehen aus einer rechten und einer linken Seite. Die rechte Seite referenziert einen Pin und

2.2 Eclipse-Sirius

Eclipse-Sirius ist ein Framework zur Erstellung von graphischen Editoren und basiert auf dem Eclipse Modelling Framework (EMF). Die Grundidee basiert auf einer logischen Trennung zwischen der semantischen Information, dem Meta-Modell, und der graphischen Repräsentation der Modellelemente durch den Editor.

2.2.1 Grundlegende Bestandteile von Sirius-Editoren

- **Sirius-Elemente**: Mapping von graphischer Repräsentation auf semantisches Element, wie z.B. Knoten oder Kanten
- **Tools**: Festlegen von verschiedenen Aktionen bzw. Verhalten des Editors z.B. das Erstellen von Knoten oder Kanten, Definieren von Doppelklicks oder Festlegen bei der Löschung eines graphischen Elements passiert
- **Properties-Ansicht:** Definition von eigenen Pages („Tabs “) für die Properties-Ansicht, beziehen sich auf Untermenge von Elementen und können z.B. genutzt werden um komplexere semantsiche Veränderungen darzustellen

AQl, Java Services

2.2.2 Diagrammerweiterungen

Ein bestehendenes Diagramm lässt sich durch Definition eines *Diagram Extension Points* erweitern. Dabei können bestehenden Mappings importiert werden und im erweiterten Editor spezialisiert bzw. modifiziert werden. Über die Viewpoint Selection kann die Erweiterung aktiviert bzw. deaktiviert werden.

3 Ergebnis

- elemente
  - Charakteristierter Prozess, Externer Aktor und Speicher importieren die jeweiligen Mappings aus dem urpsrünglichen Editor
  - Charakteristiken und Pins als Bordered Nodes
- tools
  - Hinzufügen von Pins und Charakteristiken an Knoten
  - Hinzufügen von Datenflüssen -> Anpassen der Abfragen auf
  - Verfeinerung von Prozessen: Berücksichtigen der Pins und Verhalten bei Erstellen der neuen Datenflüsse; read only Repräsentation der Charakteristiken und Pins
  - Verfeinerung von Datenflüssen
    * Verhalten von Pins
    * Erstellen von neuen Datenflüssen erfordert Erstellen von neuen Pins, dabei ist die Frage wie man mit dem Verhalten von den ursprünglichen Pins umgeht -> übertragen vielleicht nicht gewünscht, aber wie sollte es weiter repräsentiert werden?
    * benötigt Möglichkeit Verhalten zu editieren

- verhalten
  - Repräsentation der Verhalten komplexer als erwartet
  - graphische Repräsentation ungeeignet
  - textueller Editor: Xtext-Editor, bereits versucht mit Xtext-Sirius-Integration einzubinden. Injector Klasse existiert bereits, aber lies sich nicht umsetzen im zeitlichen rAhmen
- properties view
  nicht wie Diagram Extension Point definiert sondern einzelne Pages:
  - Erweitern der bestehenden properties view durch extend und override von pages
  - bei Extend entstehen zwei Tabs anstatt einem gemeinsamen
  - bei override zwar nur ein Tab aber die ursprünglichen Eigenschaften gehen nicht verloren

4 Future Work und Zusammenfassung

4.1 Future Work

- Erweiterung des Editors um einen textuellen Editor zur Bearbeitung der Verhaltenszuweisungen
- Verfeinerung von Datenflüssen zwischen den Ebenen

4.2 Zusammenfassung

Im Rahmen dieses Praktikums wurde ein bestehender Sirius-Editor für Datenflussdiagramme erweitert, um die Repräsentation von Charakteristiken von Knoten und der Verlauf der Datenfluss durch Pins. Die Verfeinerung der Prozesse und Datenflüsse wurde soweit wie möglich an das erweiterte Modell angepasst. Hierbei wurden einige konzeptionelle Herausforderungen ausgearbeitet, die sich durch das komplexere Metamodell ergeben, insbesondere bezogen auf die Zuweisungen innerhalb der Verhaltensdefinitionen.

Literatur

[1] Stephan Seifermann. *Trust40-Project/Palladio-Supporting-DataFlowDiagramConfidentiality*. `https://github.com/Trust40-Project/Palladio-Supporting-DataFlowDiagramConfidentiality`. 2020.

[2] Simon Schwarz und Stephan Seifermann. *Trust40-Project/Palladio-Supporting-DataFlowDiagram*. `https://github.com/Trust40-Project/Palladio-Supporting-DataFlowDiagram`. 2020.