

Sirius-Editor für Datenflussdiagramme

Praktikum Ingenieurmäßige Softwareentwicklung

Simon Schwarz

Betreuer: Stephan Seifermann

SOFTWARE DESIGN AND QUALITY GROUP,
INSTITUTE FOR PROGRAM STRUCTURES AND DATA ORGANIZATION, KIT DEPARTMENT OF INFORMATICS



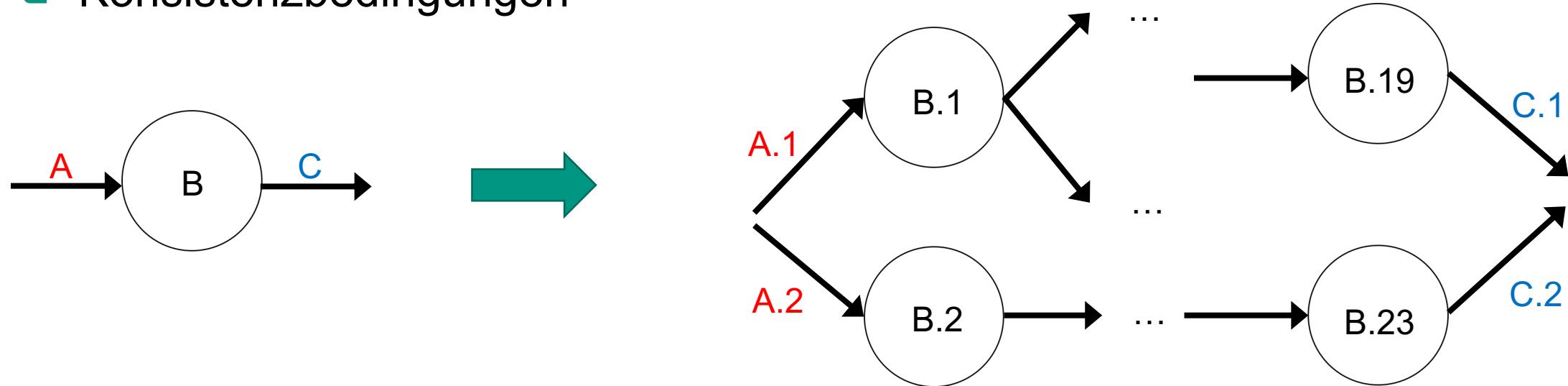
Motivation (1)

- Datenflussdiagramme: Verwendung in vielen Szenarien
 - Sicherheitsanalyse
 - Requirements Engineering
- Bisher: Kein Editor ist Industriestandard
 - Insb. kein Editor im Eclipse-Ökosystem
- Zeichenprogramme möglich, aber:
 - Semantik geht verloren
 - Keine Unterstützung bei Erstellung

Motivation (2)

■ Datenflussdiagramme: Leveling möglich

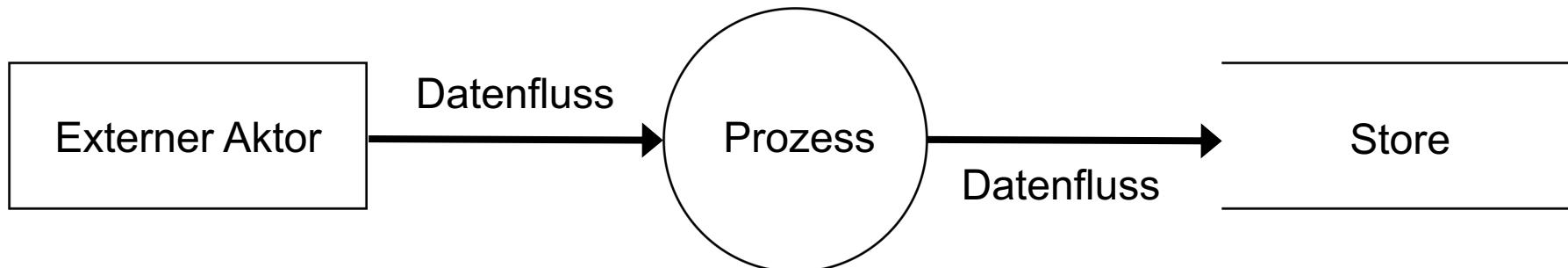
- Versch. Abstraktionslevel
- Konsistenzbedingungen



Ziel: Implementierung eines solchen Editors

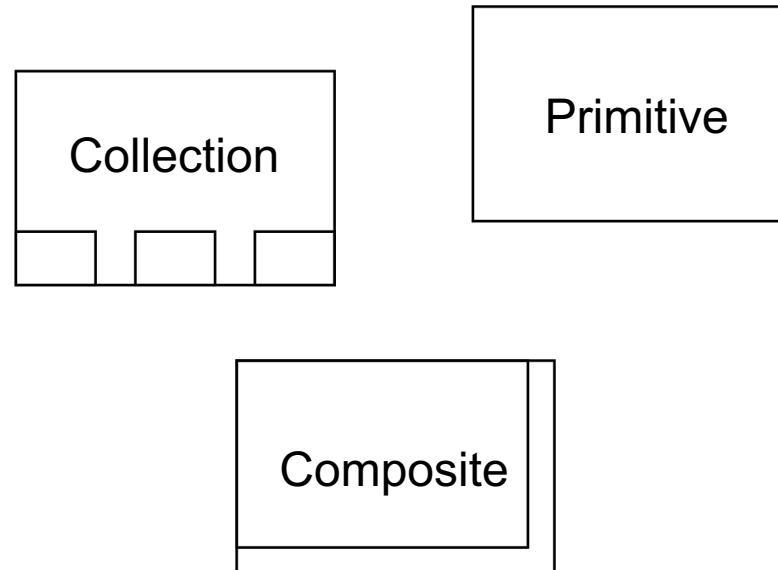
Datenflussdiagramme

- Datenorientierte Darstellung von Systemen
- 4 Einheiten
- Verfeinern in Subdiagramme möglich

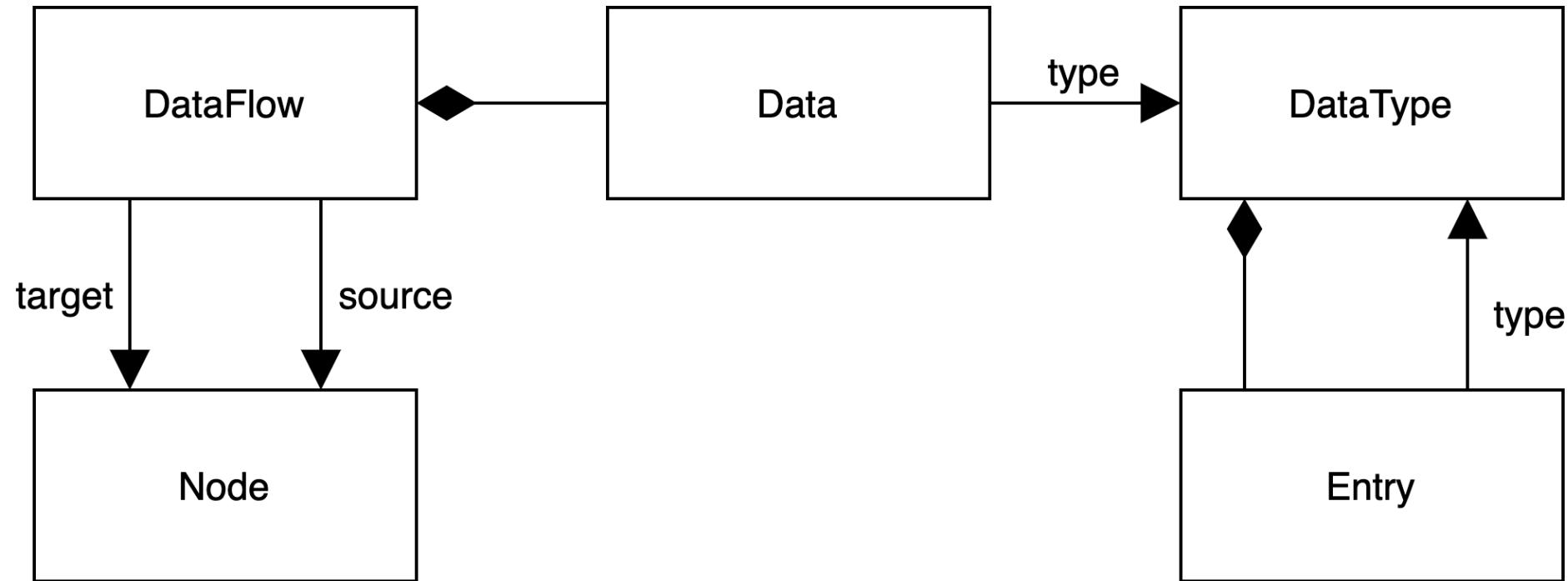


Data Dictionary

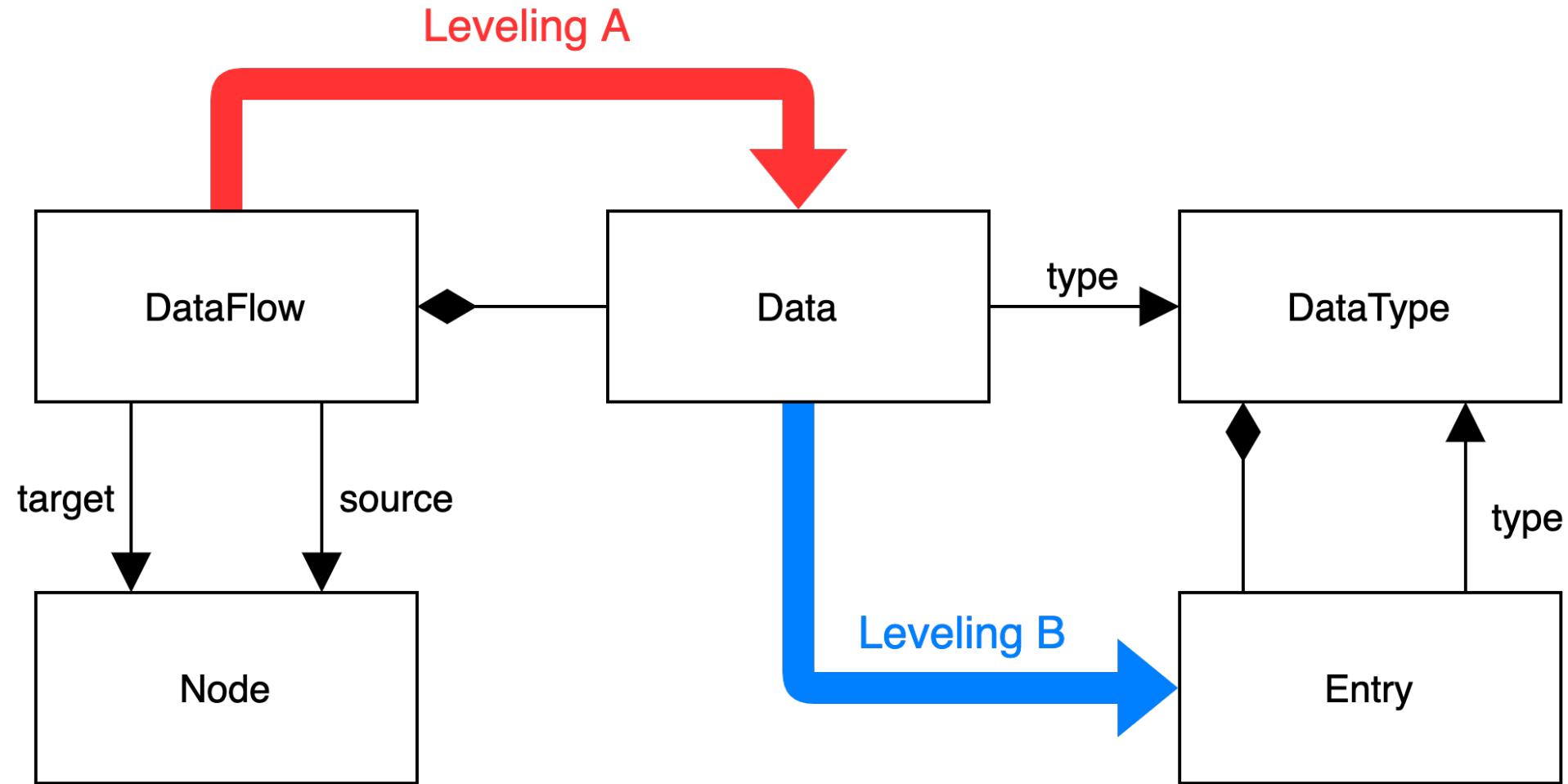
- Definition von Typkontext
 - d.h. Menge von Datentypen, die in Datenflussdiagramm vorkommen können
- 3 Einheiten
- Grundlage für
 - Leveling
 - Konsistenzbedingungen



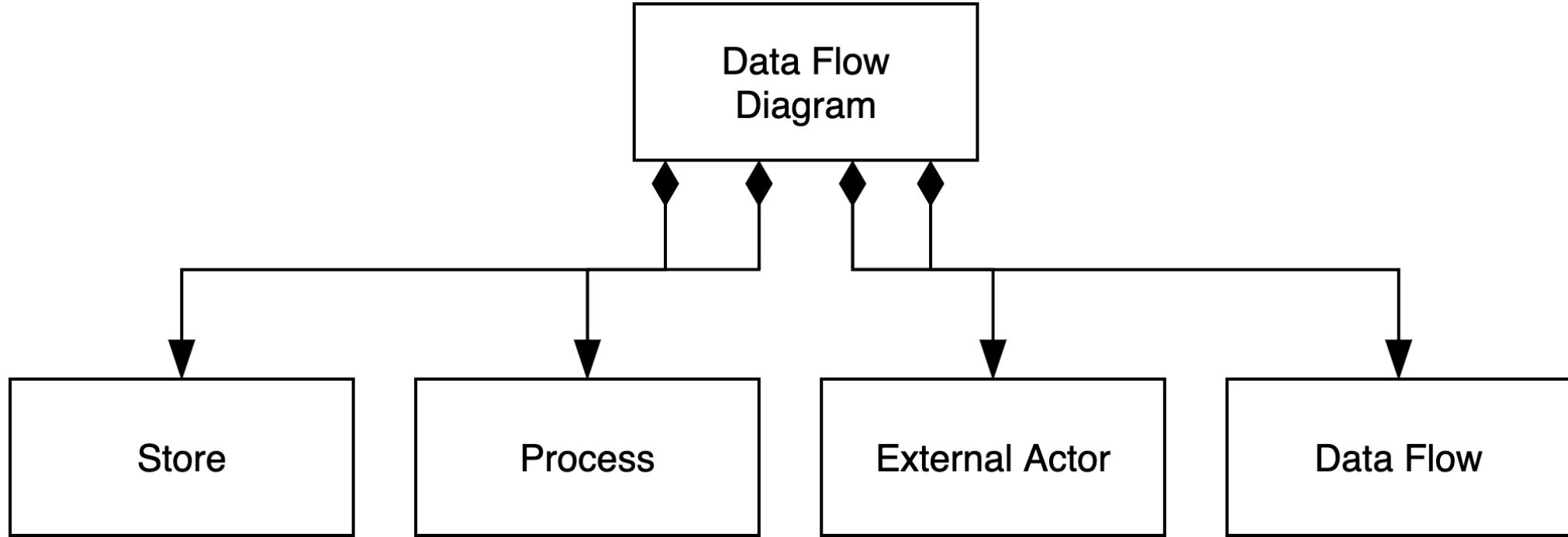
Leveling (1)



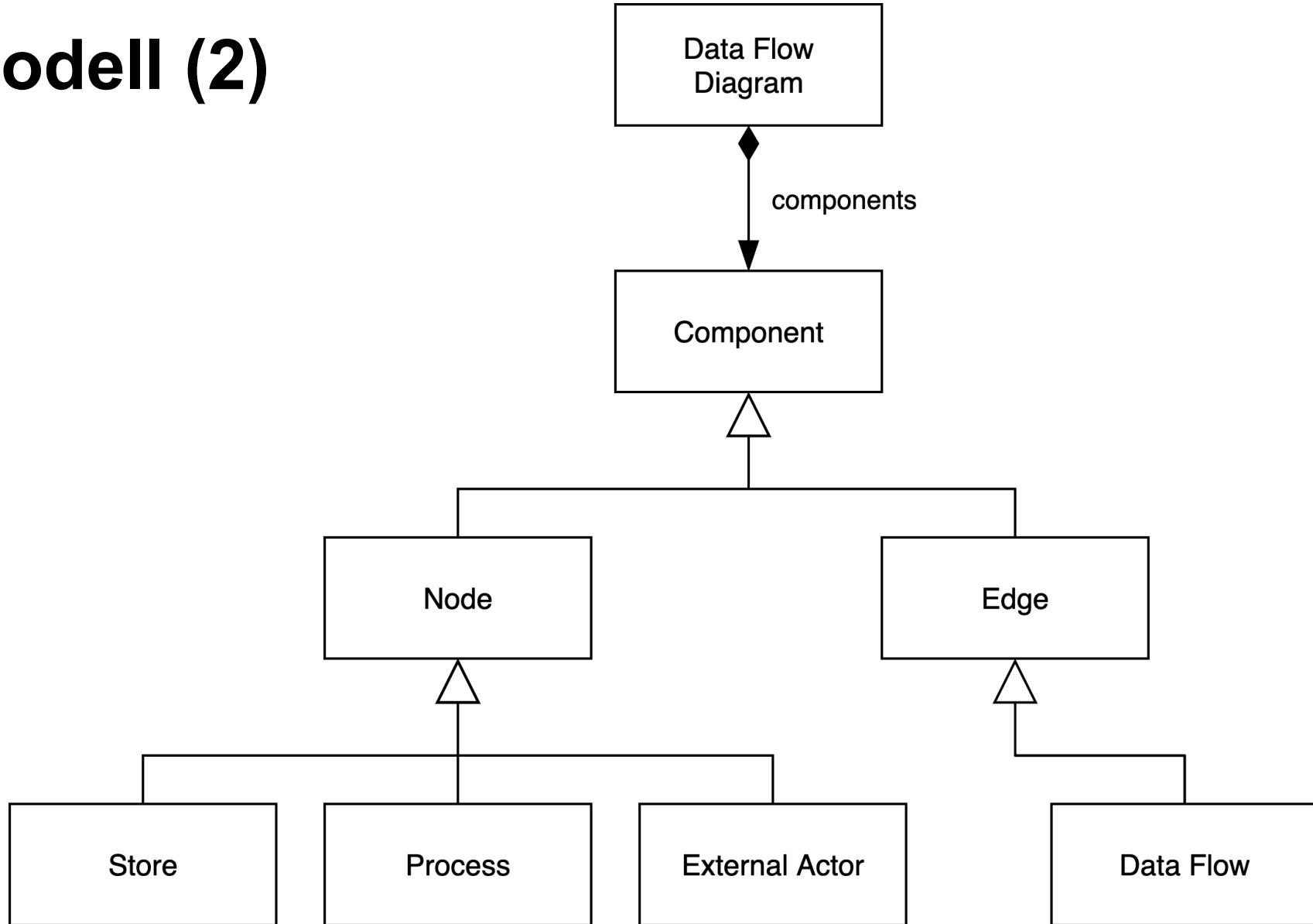
Leveling (2)



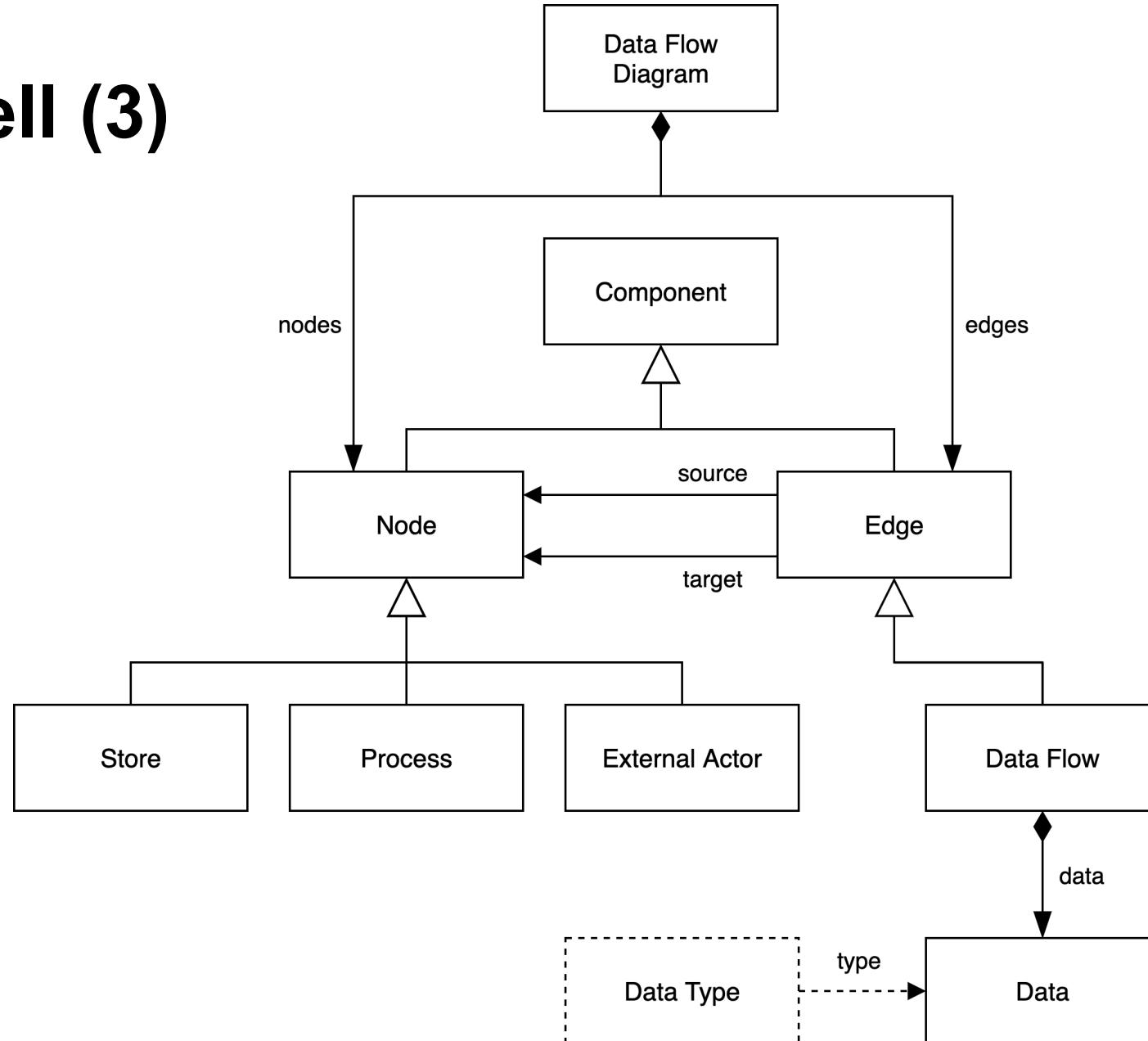
Meta-Modell (1)

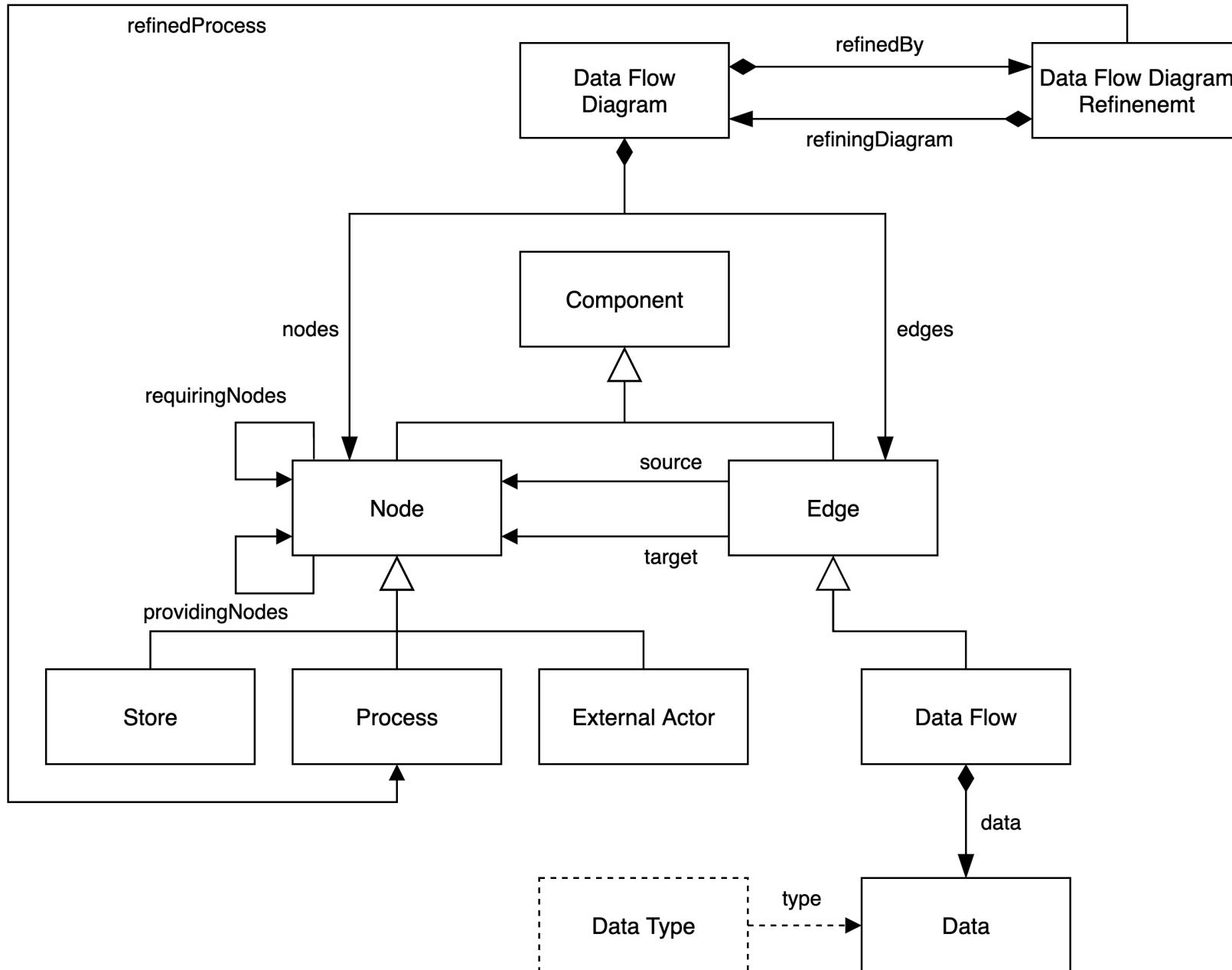


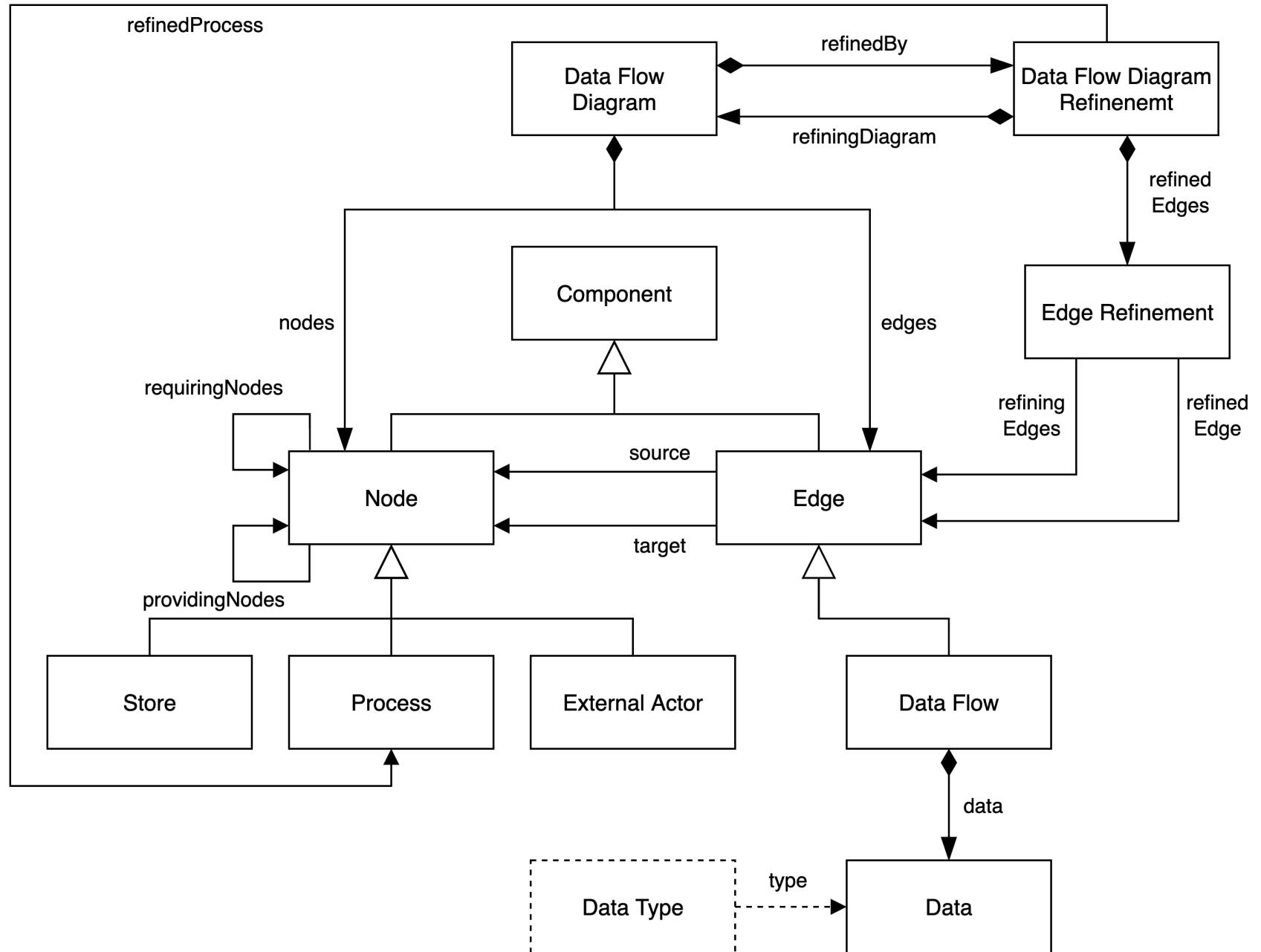
Meta-Modell (2)



Meta-Modell (3)

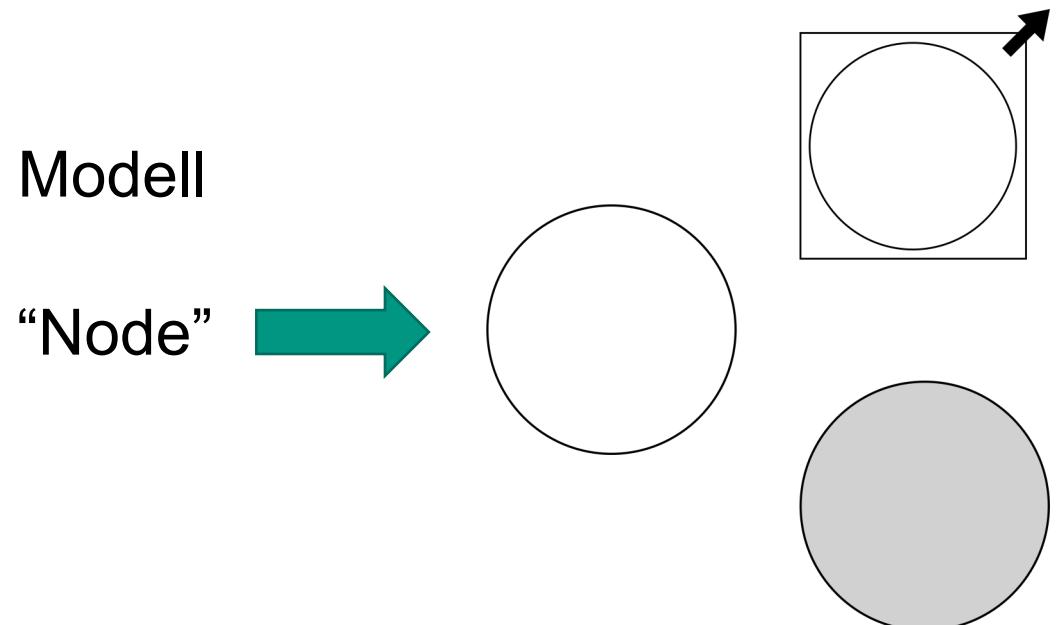






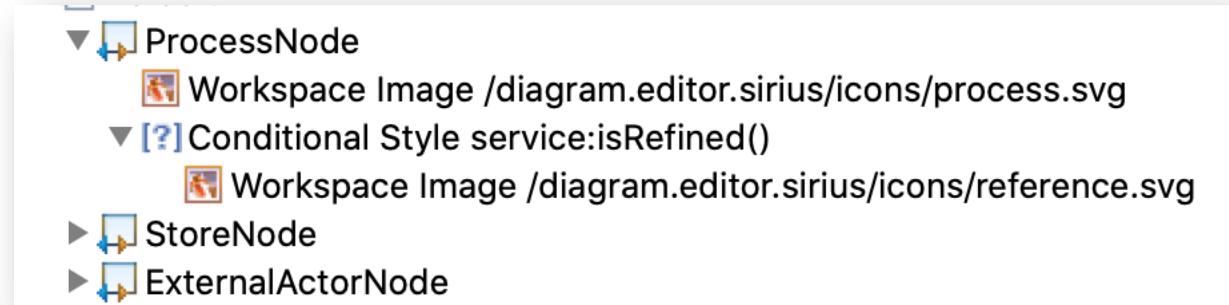
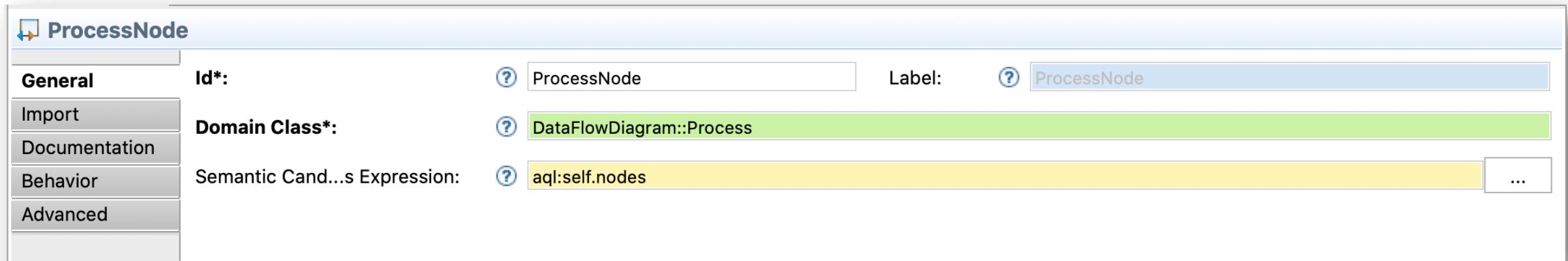
Eclipse – Sirius (1)

- Framework für Erstellung von Editoren
 - Auf Basis von Meta-Modellen
- Logische Trennung von Modell und Editor
 - Modell enthält Daten (in .xml-Datei)
 - Editor als GUI für dessen Modifizierung
 - Versch. Instanzen und Typen für *selbes* Modell



Eclipse – Sirius (2)

■ Elemente

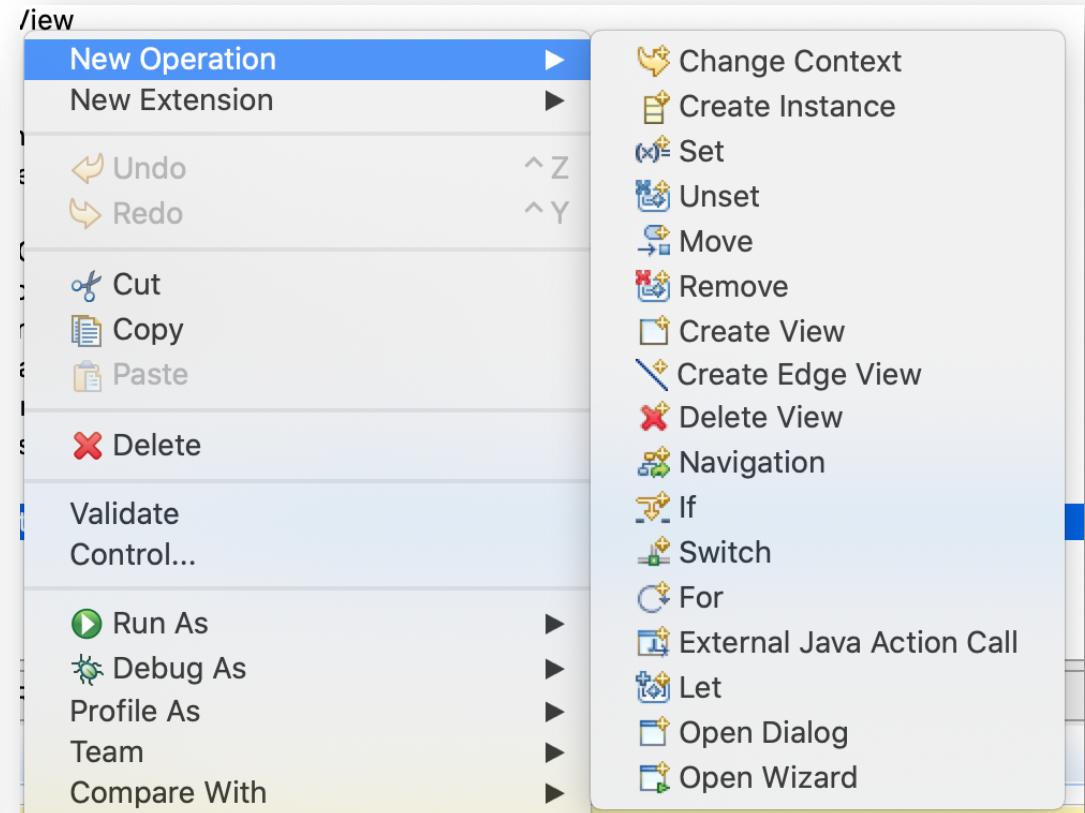
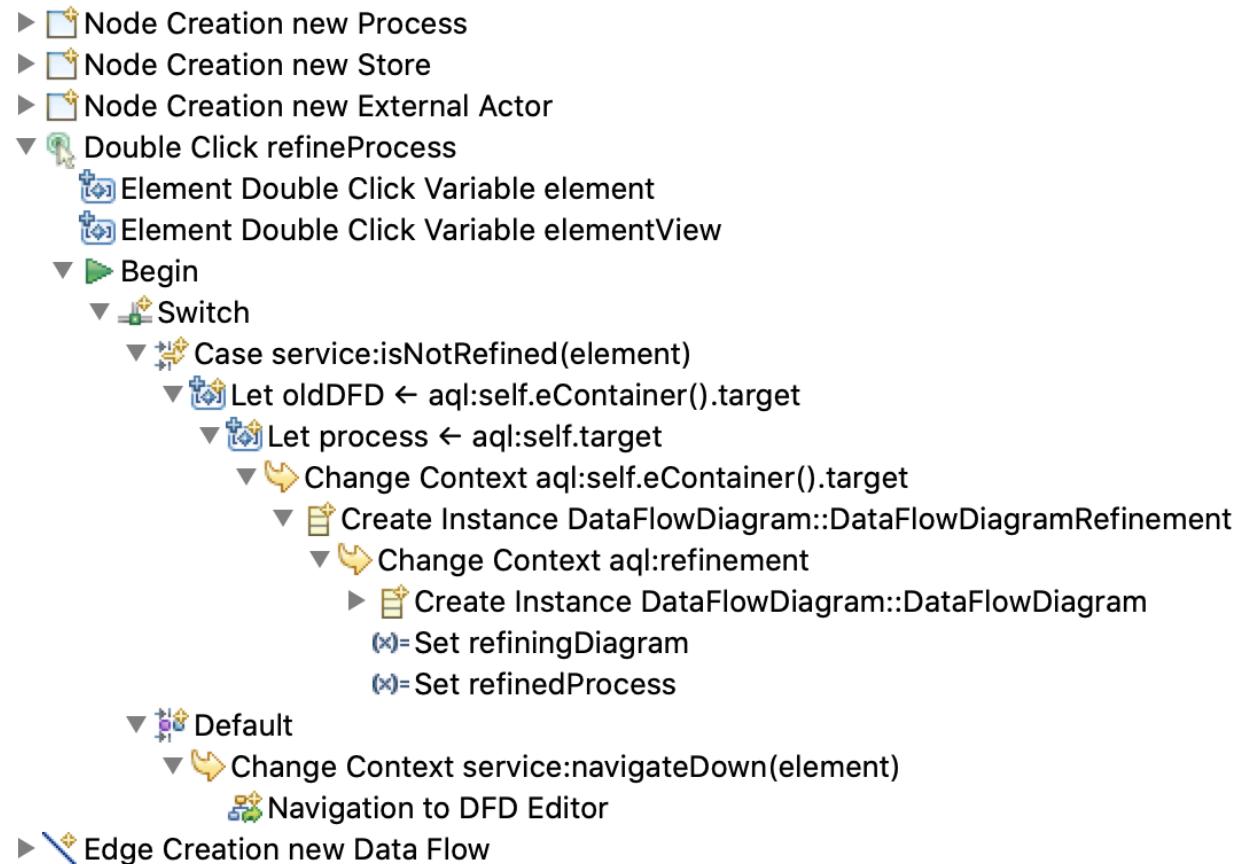
ProcessNode

General	
Id*:	(?) ProcessNode
Domain Class*:	(?) DataFlowDiagram::Process
Semantic Candidate Expression:	(?) <code>aql:self.nodes</code>

Import
Documentation
Behavior
Advanced

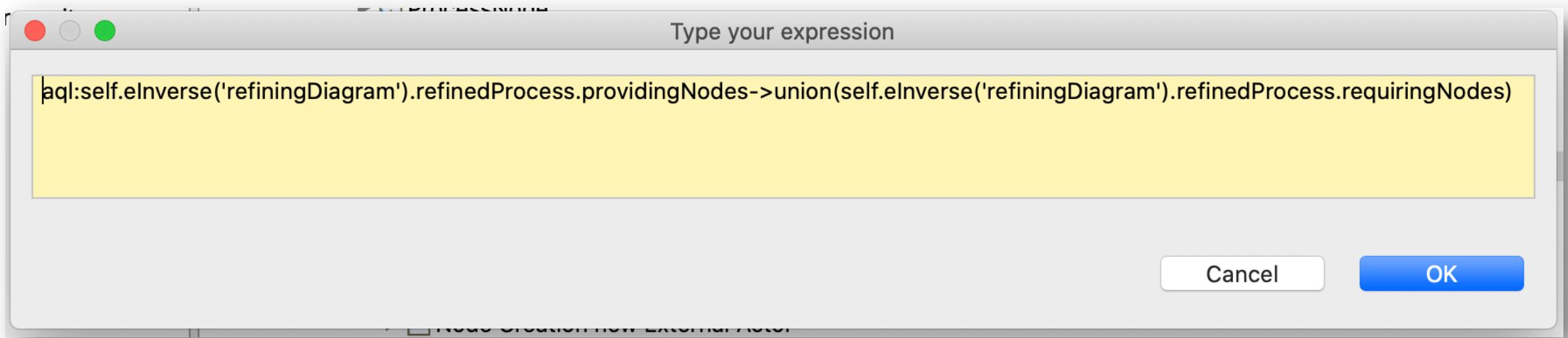
Eclipse – Sirius (3)

■ Operationen



Eclipse – Sirius (4)

■ AQL



Eclipse – Sirius (5)

■ Java-Services

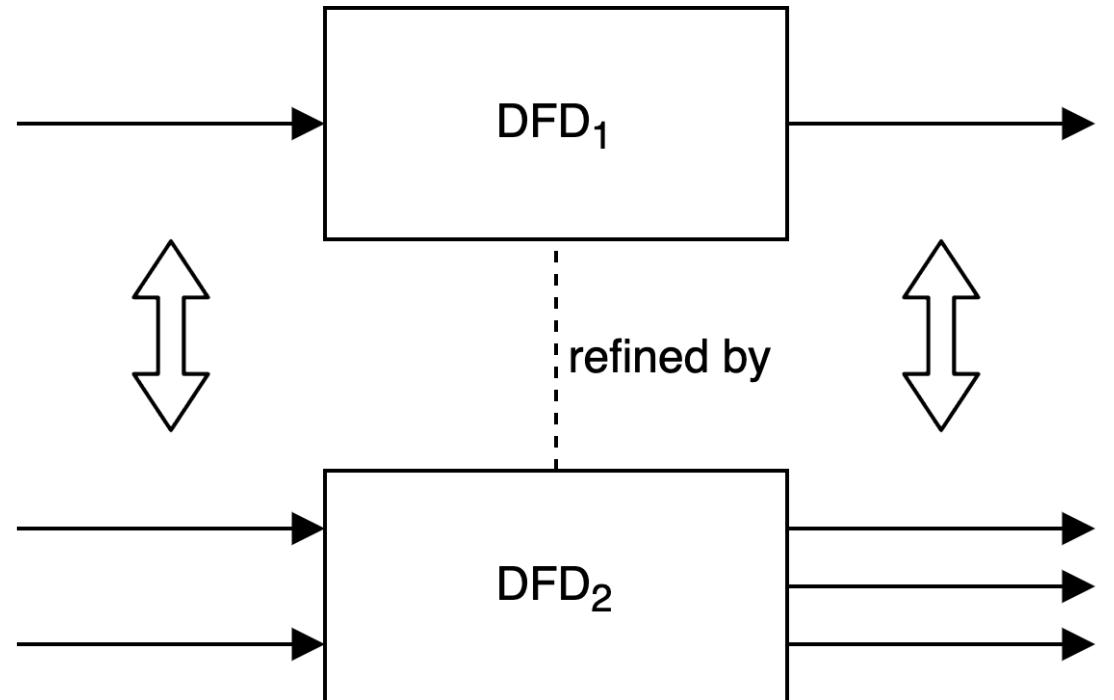
```
public void loadResources(EObject self) {
    ResourceDialog r = new ResourceDialog(Display.getCurrent().getActiveShell(), "Load Data Dictionary",
        SWT.SINGLE);
    r.open();
    r.setBlockOnOpen(true);
    Session session = SessionManager.INSTANCE.getSession(self);
    for (URI uri : r.getURIs()) {
        if (!DFDTypeUtil.uriAlreadyLoaded(uri, session))
            session.addSemanticResource(uri, new NullProgressMonitor());
    }
}
```

Validierung (1)

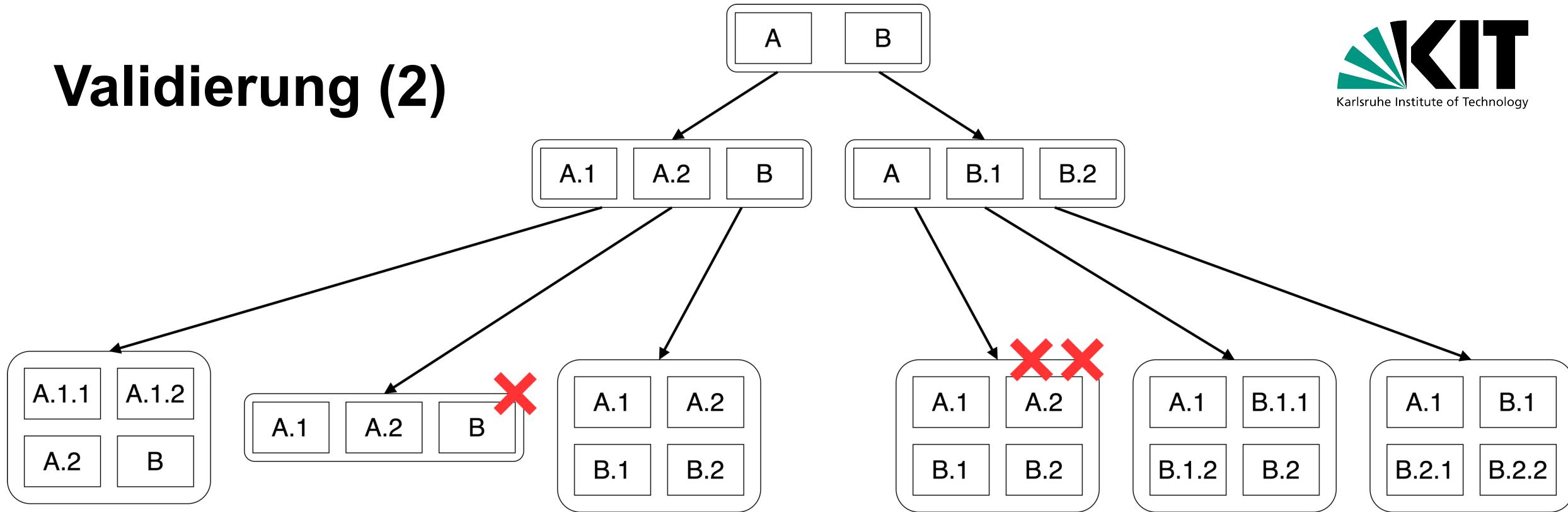
■ Frage: „Ist aktueller Zustand konsistent?“

■ Probleme:

- Automatisches Leveling
- Mapping der Datenflüsse
- Nutzereingabe



Validierung (2)

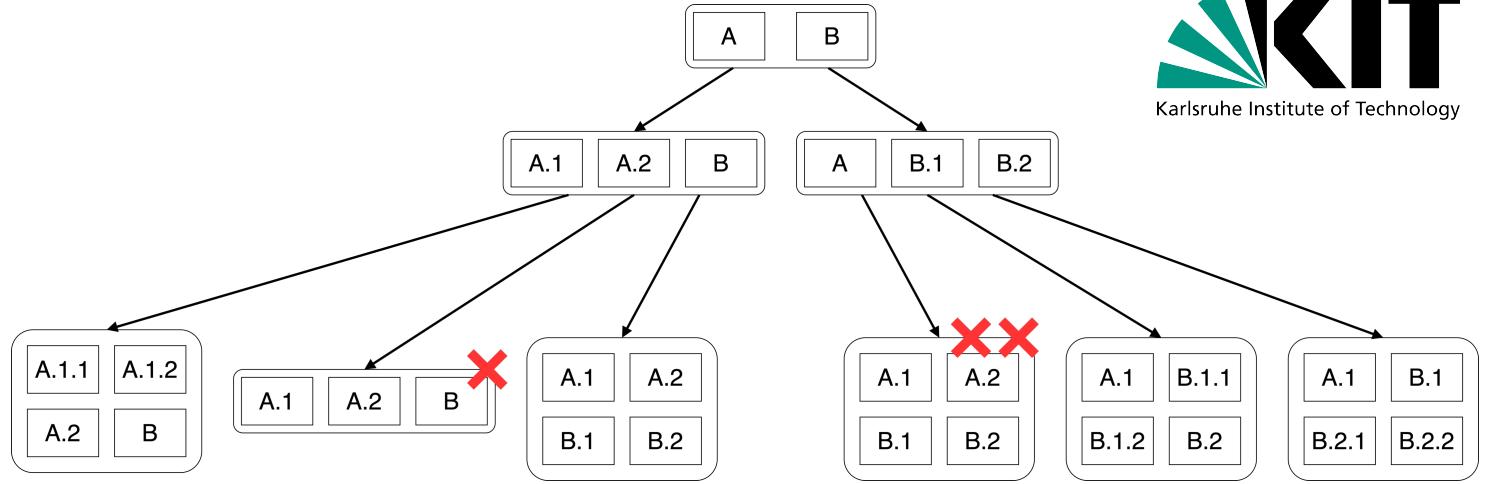


- {A.1, A.2, B} ist erlaubt
- {A.1, B} nicht
- {A.1, A.2, B.1} auch nicht
- Knoten ist Multi-Menge
- „Ausnahmen“ generieren alle Kandidaten
- Pruning von Duplikaten

Validierung (3)

■ Future Work:

- Early Pruning
- Schrittweite
- Vorberechnung
- Notwendige vs. hinreichende Kriterien
- „Greedy“-Strategie möglich?
- Erklärbarkeit



Demo

- Szenario: Teilnahme an Lehrveranstaltung

Zusammenfassung

- Datenflussdiagramme: Verwendung in vielen Teilbereichen
 - Datenorientierte Sicht
 - Versch. Abstraktionsebenen
- Bislang existierte kein geeigneter Editor
- Leveling konzeptionell schwerster Aspekt
 - Genau zu definierende Semantik
 - Validierung der Konsistenz
- Mithilfe von Eclipse Sirius: Modellorientierte Entwicklung