Projektseminar Modellbasierte Softwareentwickliung

Paul Wiedenbeck, Sebastian Ehmes, Nicolas Acero, Huyhn-Tan Truong

TECHNISCHE UNIVERSITÄT DARMSTADT

Gliederung

TECHNISCHE UNIVERSITÄT DARMSTADT

I. Szenario/Motivation

II. Überblick

III. Domain Specific Language (via Xtext)

IV. Transformation - DSL ↔ Modell

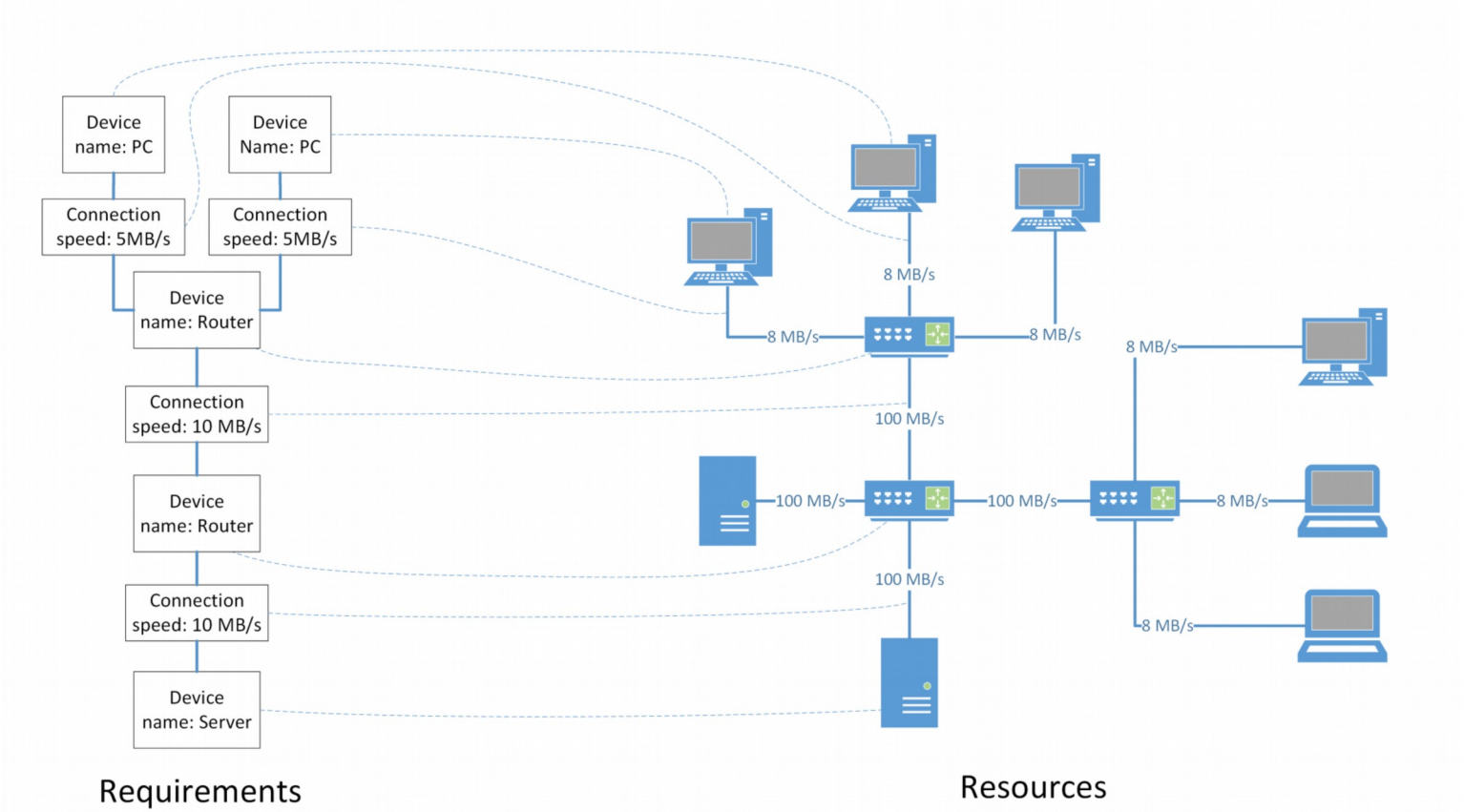
V. Transformation - Requirement ↔ Implementation

- Umsetzung 1
- Umsetzung 2

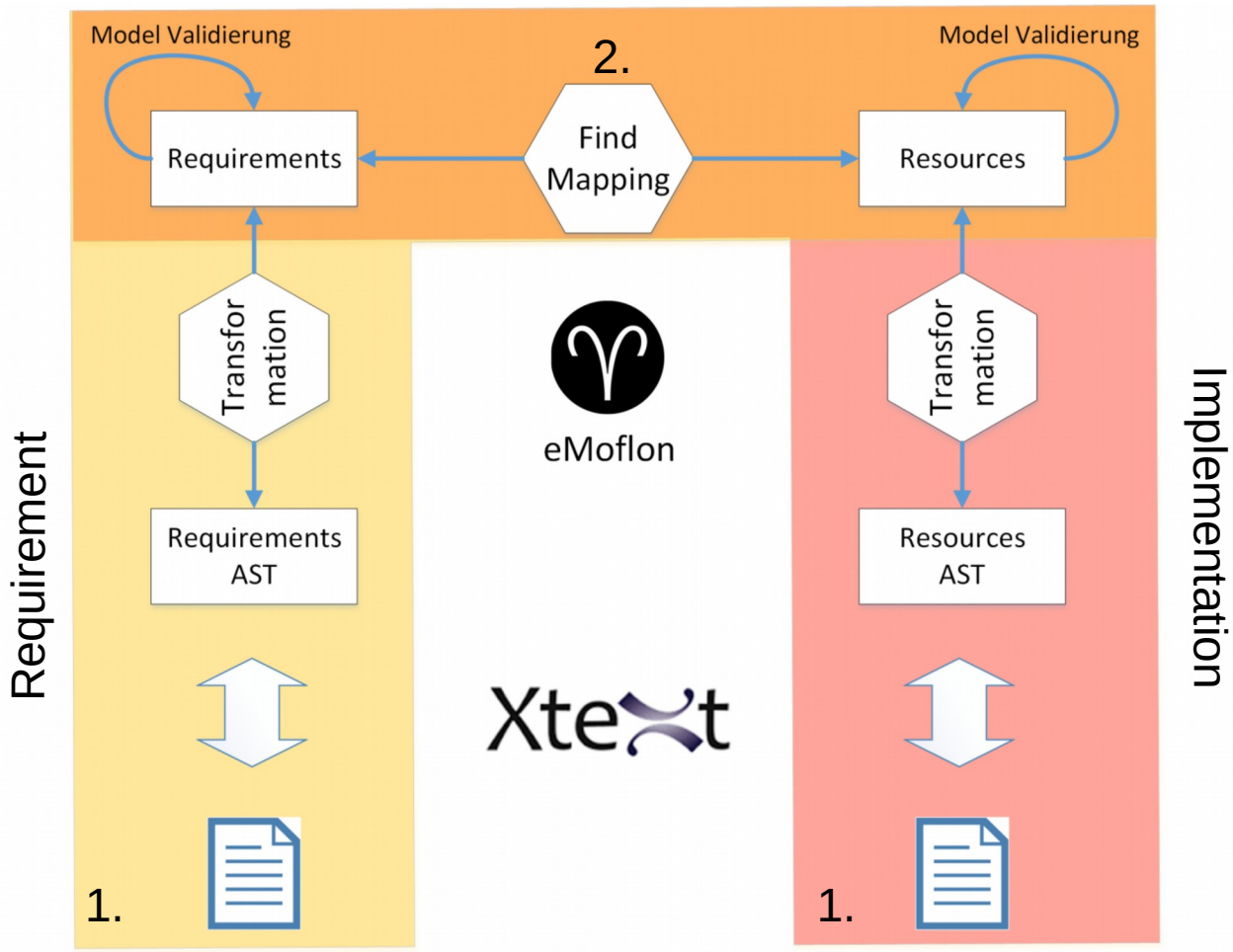
VI. Ergebnisse

I. Szenario/Motivation

- Abbildung einer **Requirement Spezifikation (Requirement)** auf reale **Ressourcen eines Netzwerks (Implementation)**



II. Überblick



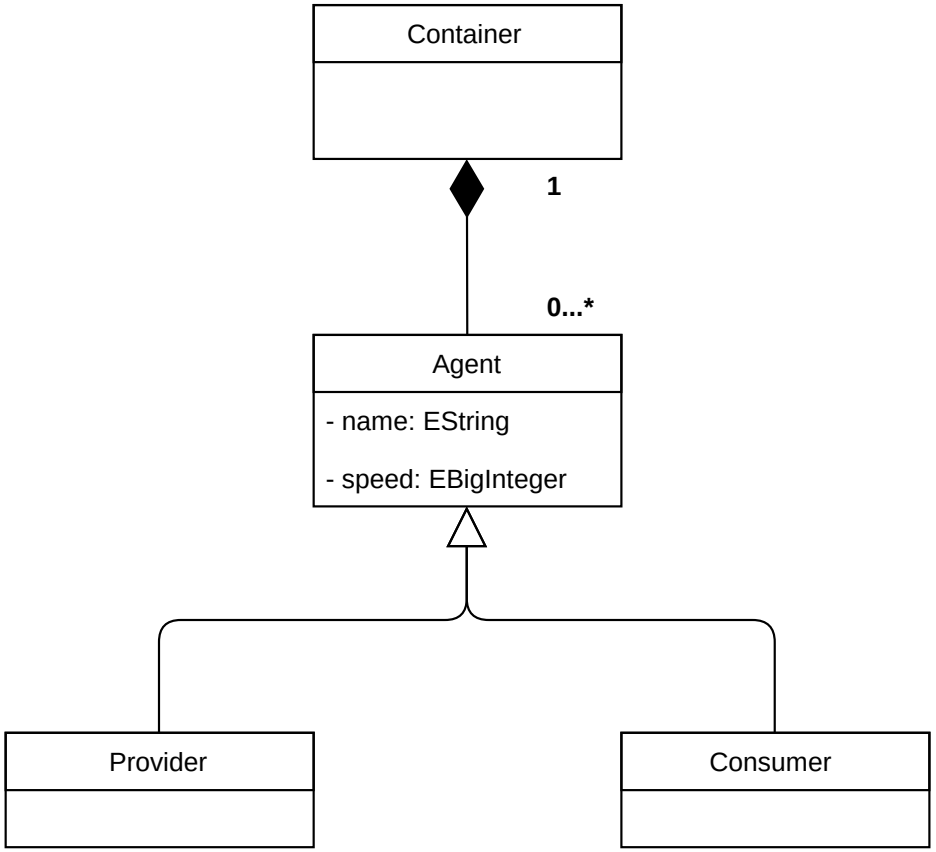
III. Xtext

TECHNISCHE UNIVERSITÄT DARMSTADT

- Framework für die Entwicklung von domänspezifischen Sprachen (**D**omain **S**pecific **L**anguage, DSL)

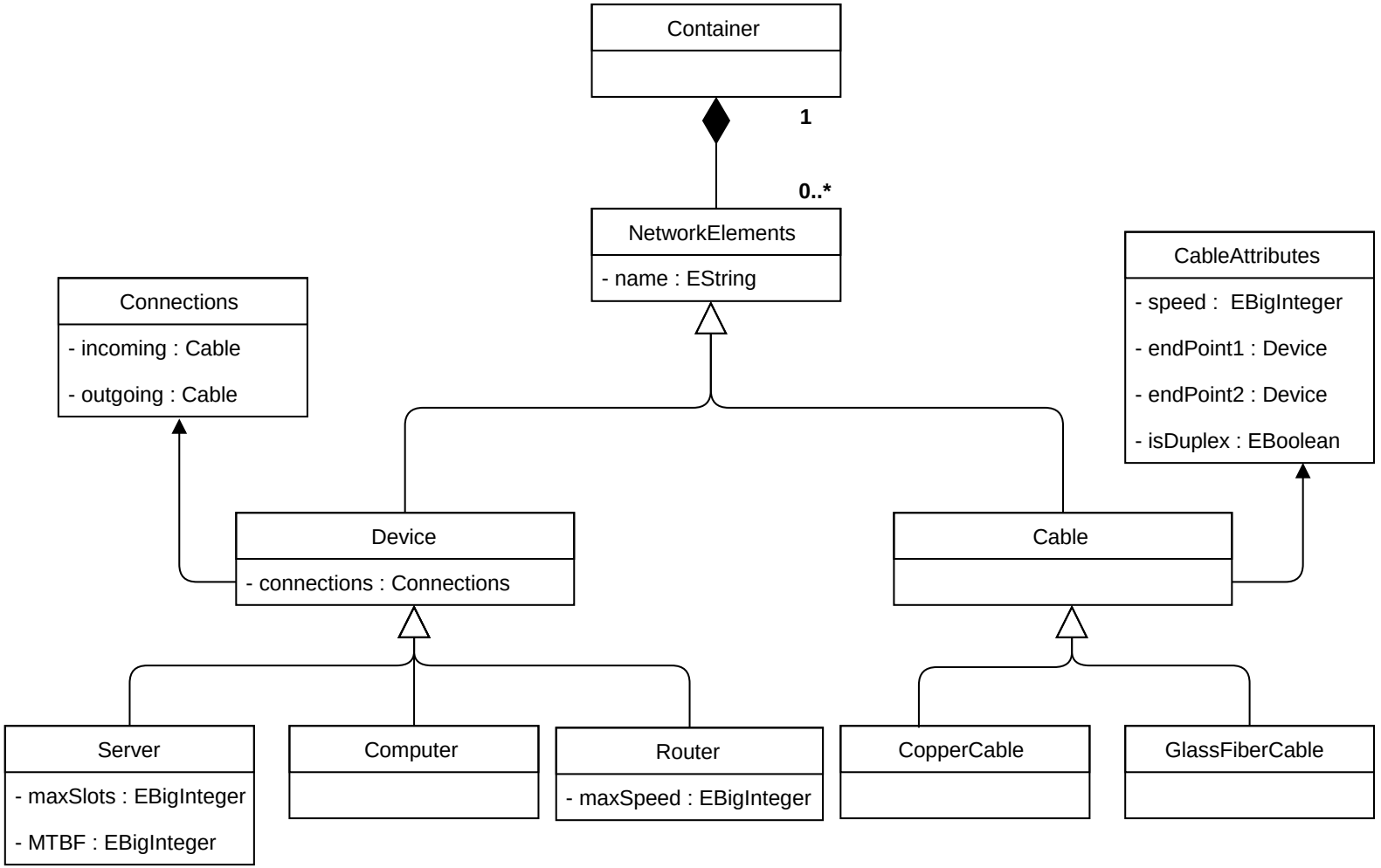
- DSLs sind Sprachen die für eine spezifische Anwendung entwickelt werden (z.B SQL für Datenbanken)

III. Entwickelte DSL - Requirement



- **Provider** können verschiedene Applikationen sein z.B. Mail-Service
- **Consumer** können verschiedene Clients sein z.B. ein User, ein Prozess
- Provider und Consumer haben Namen und haben eine bestimmte Geschwindigkeitsanforder-ung an das Netzwerk

III. Entwickelte DSL - Implementation



III. Entwickelte DSL - Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Netzwerk Elemente haben gewisse Eigenschaften
  → einhalten von Bedingungen beim Mapping von Requirements auf Implementation
- **Server:**
  - „maxSlots“ → max. Applikationen bereitstellbar
  - „MTBF“ (**M**ean **T**ime **B**etween **F**ailure) → Mittlere Betriebsdauer zwischen Ausfällen
- **Router:**
  - „maxSpeed“ → max. bereitstellbare Geschwindigkeit
- **Kabel:**
  - „speed“ → max. bereitstellbare Geschwindigkeit
  - „isDuplex“ → duplex oder simplex Verbindungen?

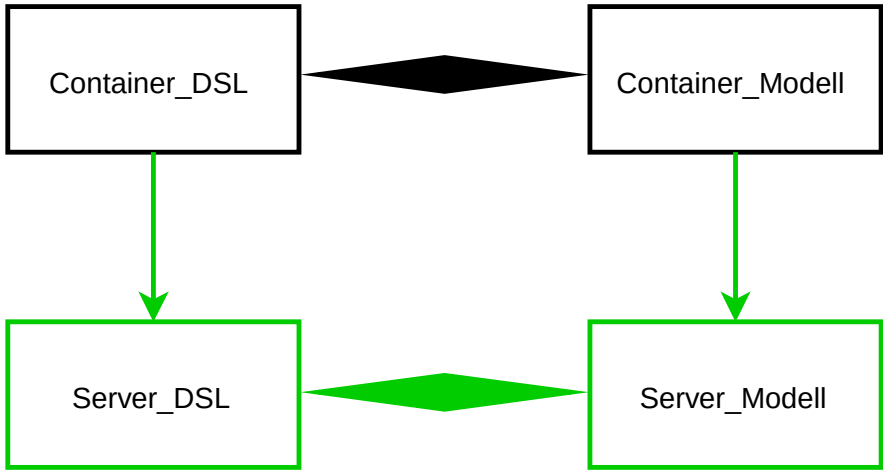
IV. Transformation – DSL ↔ Modell

TECHNISCHE UNIVERSITÄT DARMSTADT

- Transformation mit **T**riple **G**raph **G**rammars (TGGs)
- TGGs bringen zwei Graphen (**Quell- und Zielgraphen**) über einen dritten Teilgraphen (**Korrespondenzgraph**) und eine Regelmenge in Beziehung
- Diese Konsistenzspezifikation kann verwendet werden um **bidirektionale Modell-zu-Modell Transformationen** herzuleiten

IV. Transformation - DSL ↔ Modell

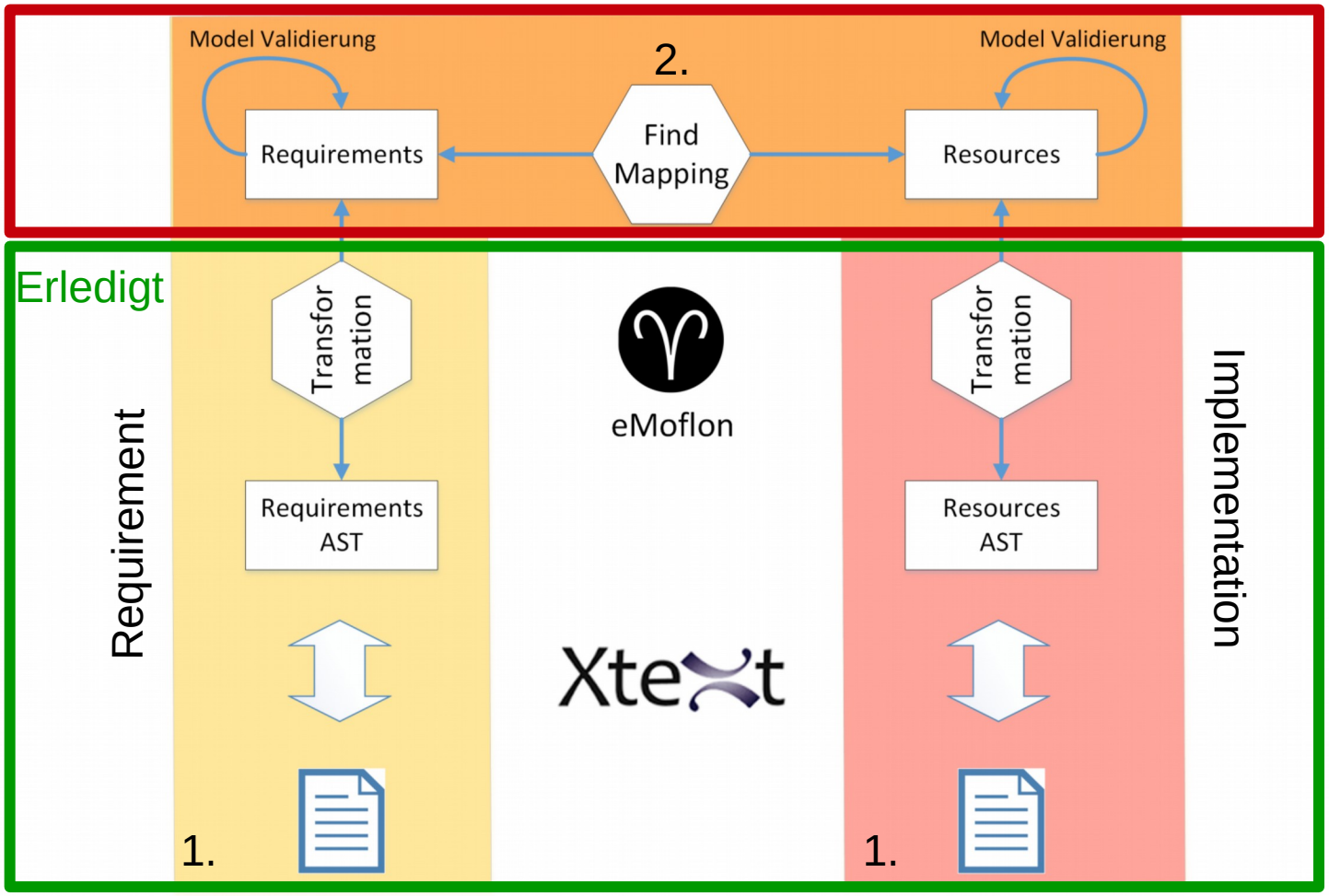
- Passende Korrespondenzen zwischen DSL und Modell definieren → **Definieren von „Regeln“**



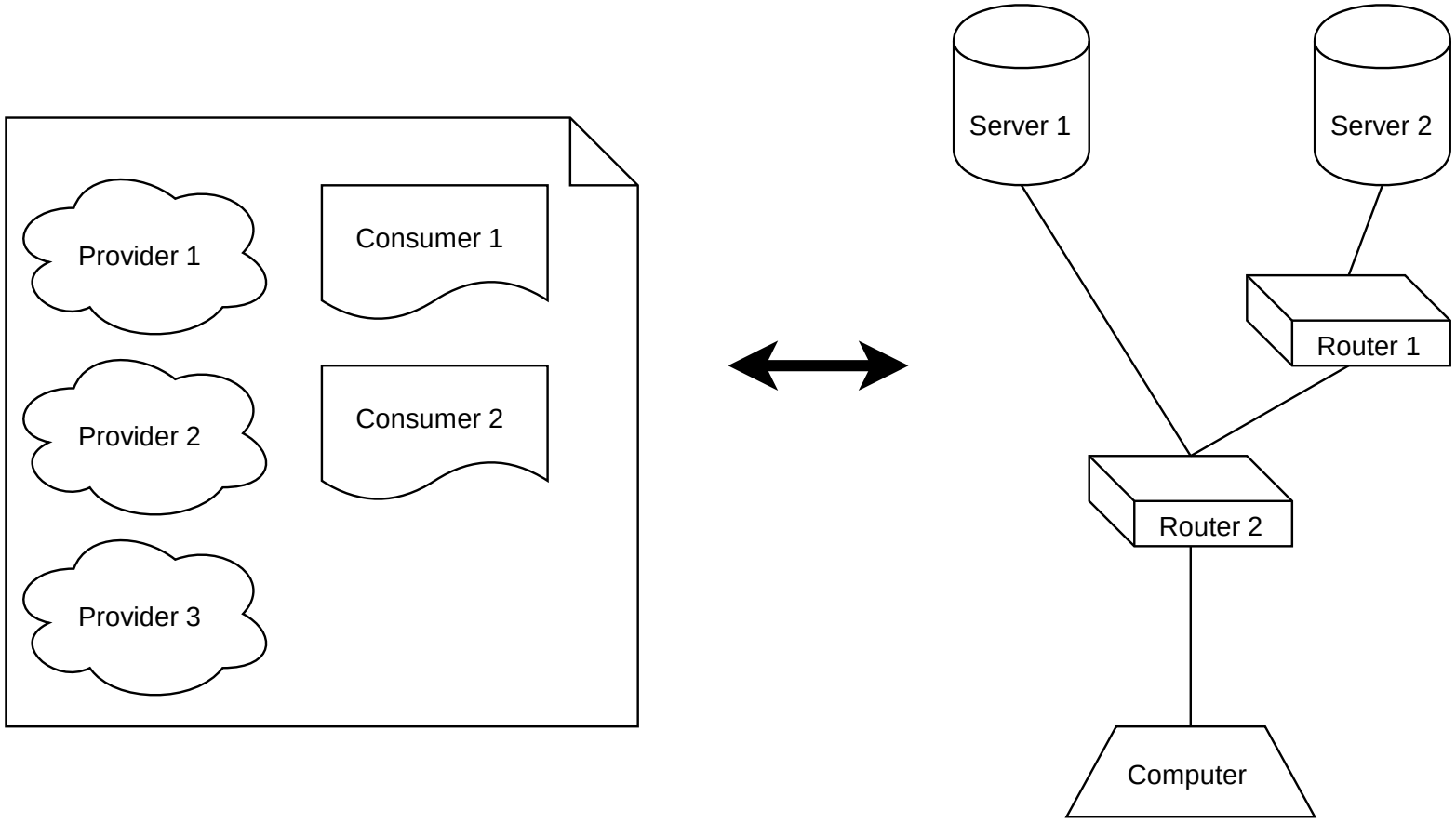
eq(Server_DSL.name, Server_Modell.name)

- Analog für die anderen Netzwerkelemente

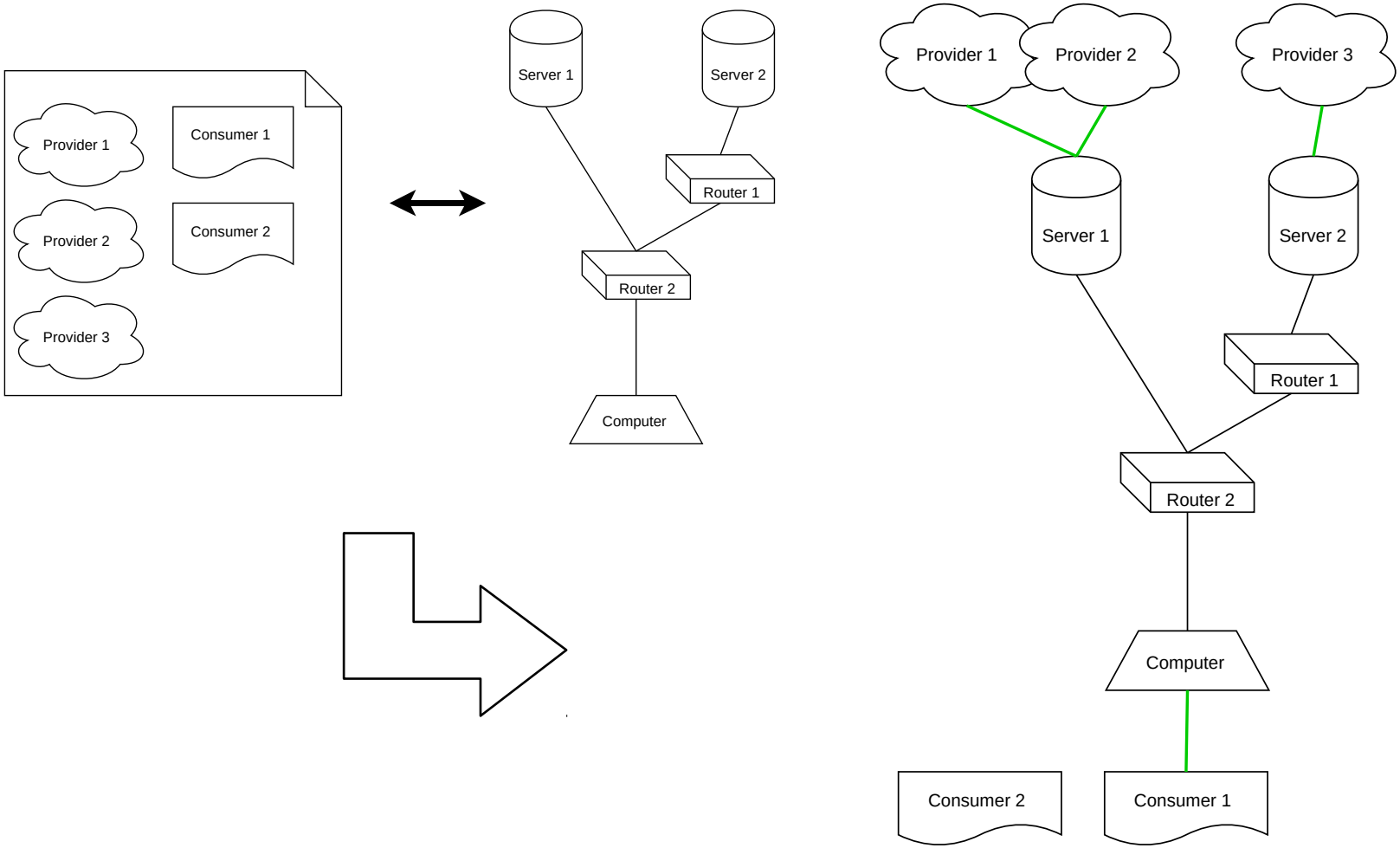
Überblick


Model Validierung
2.
Model Validierung
Requirements
Find Mapping
Resources
Erledigt
Transfor mation
Transfor mation
Requirement
eMoflon
Implementation
Requirements AST
Resources AST
Xtext
1.
1.

IV. Transformation - Requirement ↔ Implementation Beispiel



IV. Transformation - Requirement ↔ Implementation Beispiel



IV. Transformation - Requirement ↔ Implementation
Umsetzung 1 - Constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

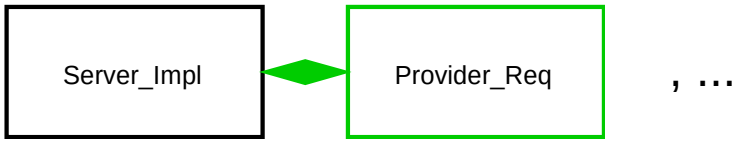
- beim Mapping müssen bestimmte Bedingung eingehalten werden:
  - ein **Server** kann nur **max. n Provider** haben
  - ein **Consumer** möchte **mindestens** eine gewisse **Downloadgeschwindigkeit** bereitgestellt bekommen
  - ein **Server** mit einer **größeren mittlere Betriebsdauer zwischen Ausfällen (MTBF)** wird bevorzugt

IV. Transformation - Requirement ↔ Implementation
Umsetzung 1

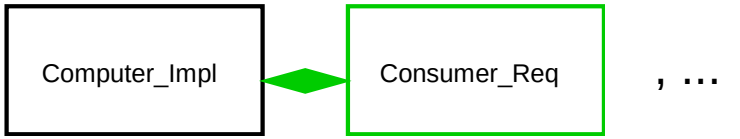
TECHNISCHE UNIVERSITÄT DARMSTADT

- **Regel 1:** Auf Implementation Seite werden alle Geräte und Verbindungen registriert

- **Regel 2:** Ohne Berücksichtigung der Nebenbedingungen werden alle Provider auf jeden Server gemappt



- **Regel 3:** Ohne Berücksichtigung der Nebenbedingungen werden alle Consumer auf jeden Computer gemappt



- Über **„Integer Linear Programming (ILP)“** werden Matches verworfen, die die Bedingungen nicht erfüllen

IV. Transformation - Requirement ↔ Implementation
Umsetzung 1 - ILP

- Beim ILP wird eine **Zielfunktion** (Objectivefunction) **maximiert**:

  $f(v_i) = \alpha v_1 + \beta v_2 + \gamma v_3 + \ldots$

- Wobei v_i 0 oder 1 ist und α, β, γ, … Gewichte sind
- z.B „Server können nur max. n Provider haben“: α, β, γ,… = 1, da alle gleichwertig sind

  → $v_1 + v_2 + v_3 + \ldots \leq n$, wird maximiert

IV. Transformation - Requirement ↔ Implementation
Umsetzung 2 - Constraints

TECHNISCHE UNIVERSITÄT DARMSTADT

- beim Mapping müssen bestimmte Bedingung eingehalten werden:
  - ein **Server** kann nur **max. n Provider** haben
  - ein **Consumer** möchte **mindestens** eine gewisse **Downloadgeschwindigkeit** bereitgestellt bekommen
  - ein **Server** mit einer **größeren mittlere Betriebsdauer zwischen Ausfällen (MTBF)** wird bevorzugt
  - ein **Server** mit **wenigen „Hops“** (kürzesten Pfad) bis zu einem Computer wird bevorzugt

IV. Transformation - Requirement ↔ Implementation
Umsetzung 2 - Idee

TECHNISCHE UNIVERSITÄT DARMSTADT

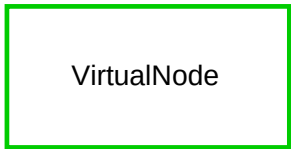
- Das implementierte Netzwerk wird auf einen „Dummy Knoten“ (**VirtualNode**) gemappt
- Beim Mappen werden die **„Hops“** (der Pfad von einem Device zu einem anderem Device) **negativ gewichtet**
- Objectivefunction beim ILP wird **maximiert**

→ Ergebnis ist der **kürzeste Pfad** von einem Server zu Computer

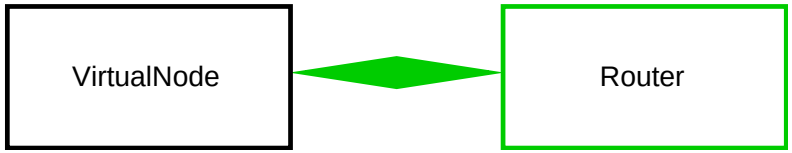
IV. Transformation - Requirement ↔ Implementation
Umsetzung 2

- zusätzlich zur Umsetzung 1, gilt es einen möglichst kurzen Pfad von Server zu Computer zu finden

- **VN_Regel 1:** Registrierung des VirtualNodes

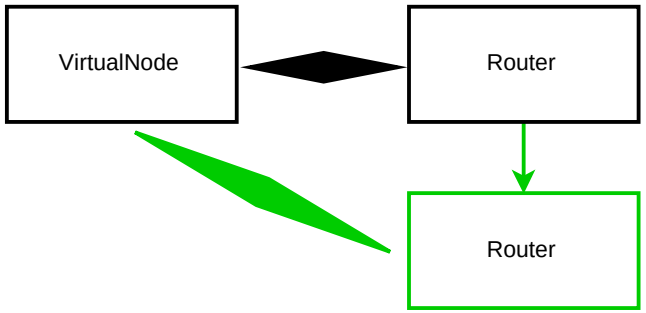


- **VN_Regel 2:** Korrespondenz zwischen VirtualNode und einem Router (wird genau einmal ausgeführt)

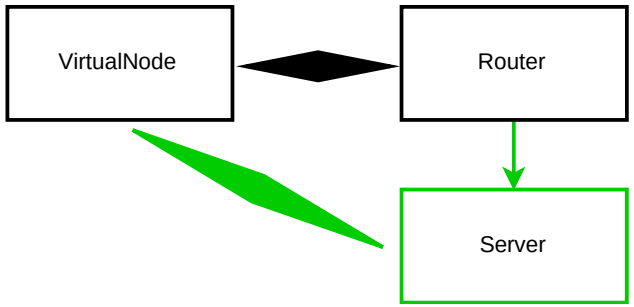


IV. Transformation - Requirement ↔ Implementation
Umsetzung 2

- **VN_Regel 3:** Korrespondenz zwischen VirtualNode und einem Router, der mit einem registrierten Router verbunden ist (diese Korrespondenz erhält ein negatives Gewicht)

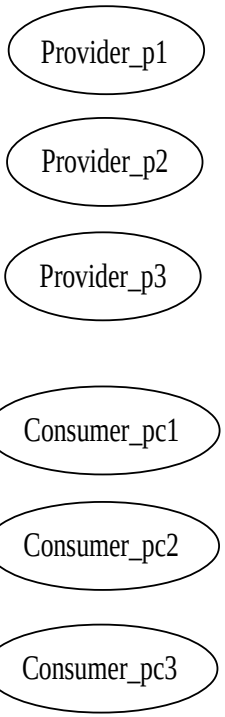


- **VN_Regel 4:** Korrespondenz zwischen VirtualNode und einem Gerät (Server/Computer), der mit einem registrierten Router verbunden ist



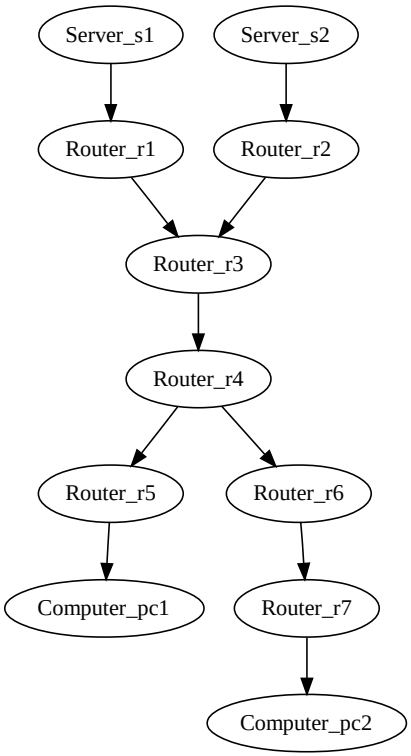
VI. Ergebnisse & Fazit

```
Provider p1 {
    speed 3
}

Provider p2 {
    speed 3
}

Provider p3 {
    speed 3
}

Consumer c1 {
    speed 2
}

Consumer c2 {
    speed 2
}

Consumer c3 {
    speed 2
}
```



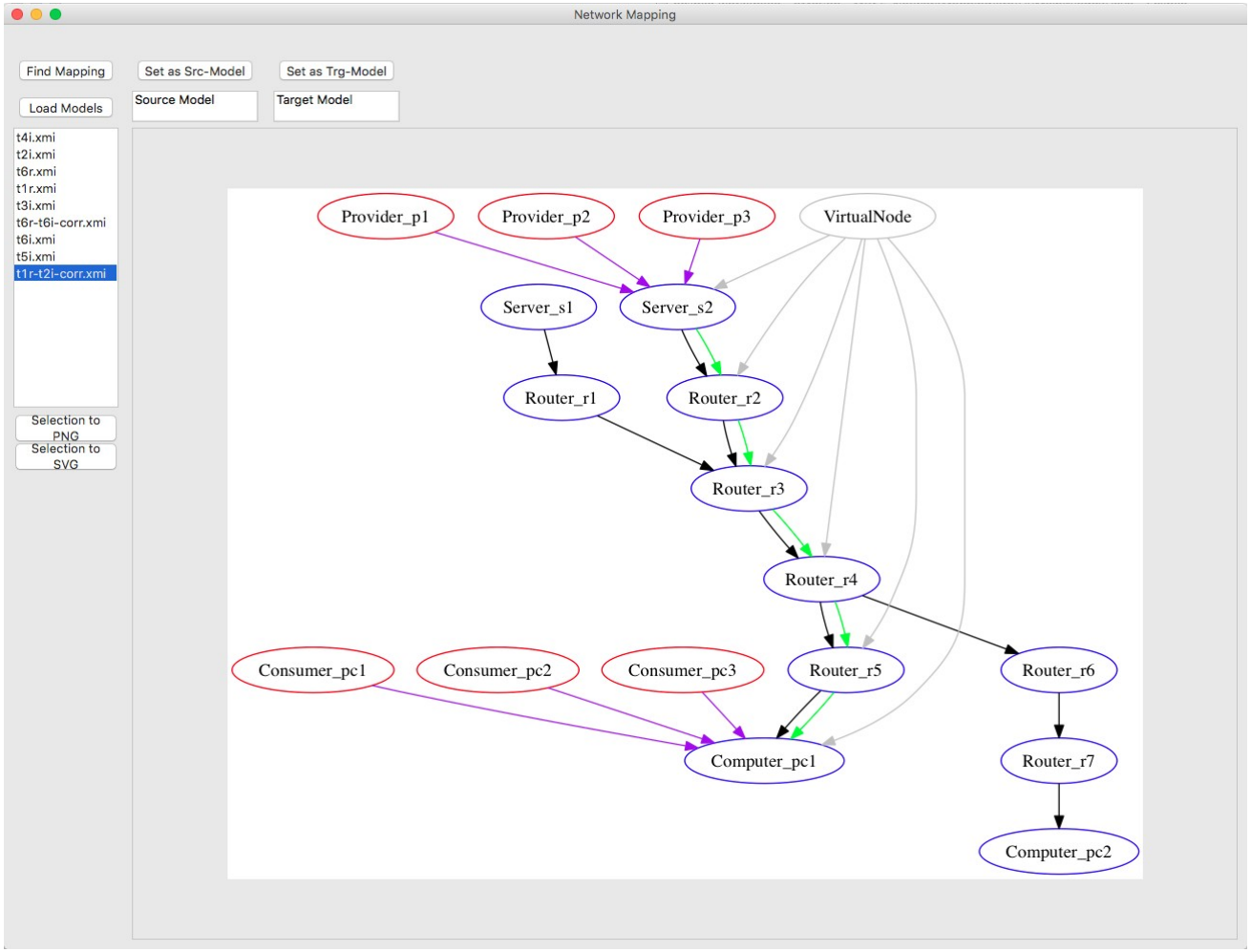
Requirement

```
Server s1{
    maxSlots 10
    MTBF 100
    incoming(s_r)
    outgoing(s_r)
}
Router r1{
    maxSpeed 100
    incoming(s1_r1)
    outgoing(r1_r3)
}
Router r3{
    maxSpeed 100
    incoming(r1_r3)
    outgoing(r3_r4)
}
GlassFiberCable s1_r1 {
    speed 100
    endPoint1 s1
    endPoint2 r1
    isDuplex false
}
GlassFiberCable r1_r3 {
    speed 100
    endPoint1 r1
    endPoint2 r3
    isDuplex false
}
```



Implementation

VI. Ergebnisse & Fazit



- Über VirtualNode kann der kürzeste Pfad abgelesen werden:
  - ➔ Alle Provider werden auf Server_s1 gemappt
  - ➔ Alle Consumer werden auf Computer_pc1 gemappt

VI. Ergebnisse & Fazit

- DSL entwickelt, um die Modelle „Requirement“ und „Implementation“ zu beschreiben
- Transformation der DSL-Modelle zu Metamodellen
- Umsetzung 1: Mappen von Requirements (Provider/Consumer) zu Implementation-Netzwerktopologien unter Nebenbedingung
- Umsetzung 2: Erweiterung der Umsetzung 1 mit zusätzlichem Finden des kürzesten Pfades für voll verbundene Topologien

Vielen Dank
Für
Ihre Aufmerksamkeit