Modellierung von Echtzeitsystemen

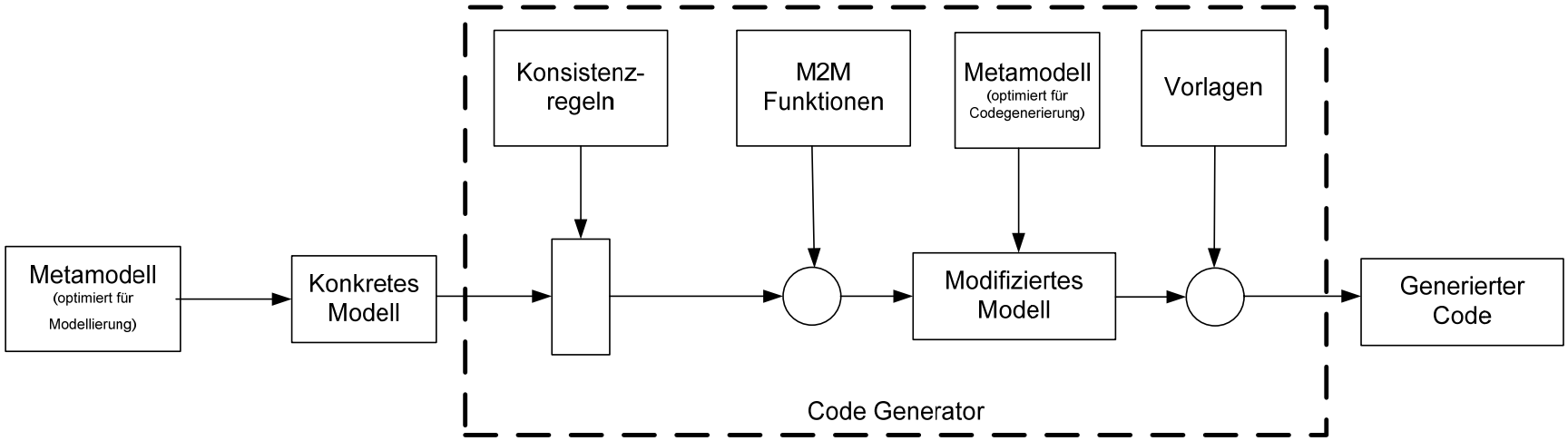
Entwicklung von Domänenspezifischen Codegeneratoren

Beispiel: FTOS

Domänenspezifische Codegeneratoren

- Beobachtung: Viele existierende Werkzeuge beschränken sich auf die Generierung der Anwendungsfunktionalität
- Grund: es kann plattformunabhängiger Code (z.B. ANSI-C) generiert werden
- Die Interaktion mit der zugrunde liegenden Hardware wird von wenigen Codegeneratoren unterstützt
  ⇒ Trend zu domänenspezifischen Codegeneratoren
- Domänenspezifische Codegeneratoren
  - Konzentrieren sich auf Konzepte/Mechanismen der Anwendungsdomäne
  - Zeichnen sich häufig durch gute Erweiterbarkeit aus
  - Können als eine Weiterentwicklung von komponentenorientierten Entwicklungsansätzen interpretiert werden
- Es existieren gute Infrastrukturen zum schnellen Erstellen von solchen Entwicklungswerkzeugen, z.B. openArchitectureWare, metaEdit, GME

Bestandteile eines Codegenerators



Beschreibung der Konzepte

Modellierung (durch den Benutzer)

Konsistenz-prüfung

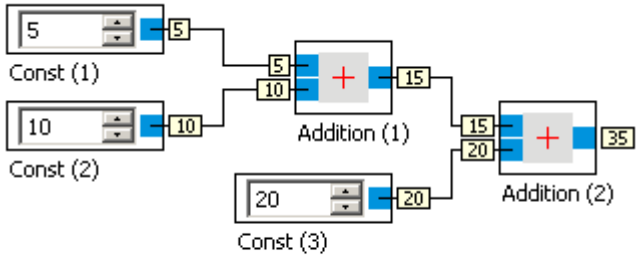
Modell Transformation

Codegenerierung

Bestandteile eines Codegenerators

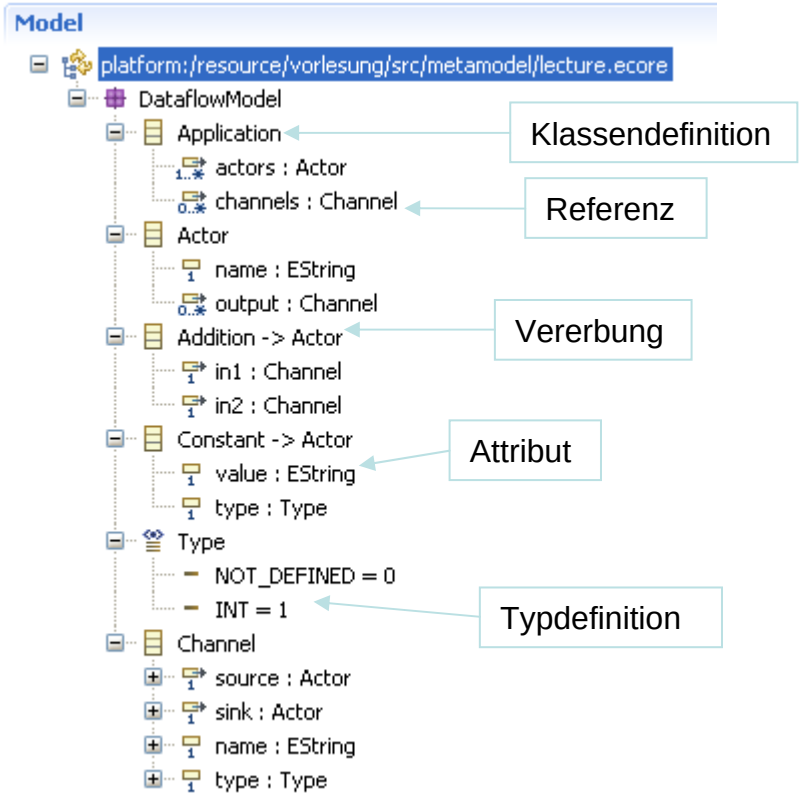
- Metamodelle:
  - Das Metamodell definiert die Modellierungskonzepte und die Zusammenhänge
  - Auf Basis des Metamodells werden Konsistenzregeln, Modell-zu-Modelltransformationsregeln, sowie die Codegenerierungsregeln definiert
- Konkretes Modell
  - Das konkrete Modell durch den Entwickler basiert auf dem Metamodell und beschreibt die konkrete Anwendung
- Konsistenzregeln:
  - Die fehlerfreie Modellierung wird durch entsprechende Konsistenzregeln geprüft
- Modell-zu-Modell-Transformation:
  - Durch eine Modell-zu-Modell-Transformation können mehrere Modelle zusammengefasst und die leichte Modellierung, sowie Codegenerierung gewährleistet werden
  - Die Modell-zu-Modell-Transformation überführt ein Modell basierend auf einer für eine einfache Modellierung optimiertes Metamodell in ein Modell basierend auf ein für die Codegenerierung optimiertes Modell
- Vorlagen:
  - Vorlagen beschreiben die Regeln zur Generierung von Code auf der Basis eines konkreten Modells

Beispiel: Codegenerator für Synchronen Datenfluss



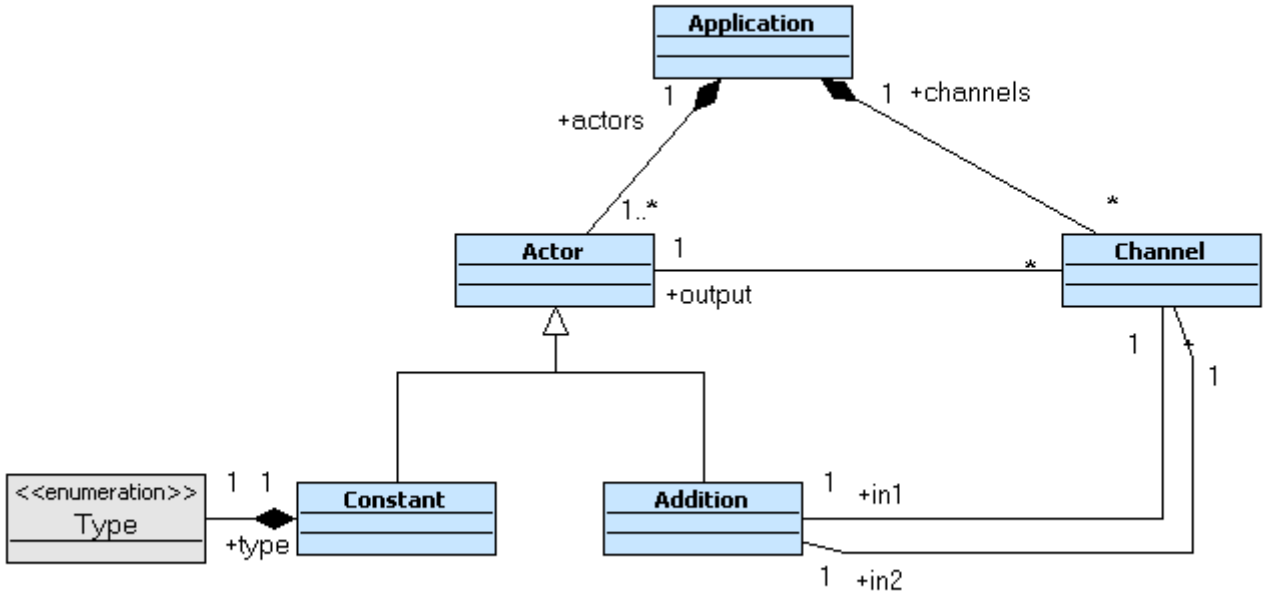
- Einschränkungen zur Vereinfachung:
  - Jeder Aktor hat nur einen Ausgang namens output
  - Maximal ein Aktor darf einen nicht verbundenen Ausgang besitzen ⇒ Ergebnis
- Verwendung von openArchitectureWare als Codegenerierungsinfrastruktur http://www.openarchitectureware.org/
- Download Komplettinstallation: http://www.gentleware.com/oaw.html
- Den Codegenerator können Sie auf der Vorlesungsseite herunterladen

Metamodelle



- Das Metamodell definiert, wie eine Anwendung beschrieben werden kann
- Zur Definition der Metamodelle werden in der Regel Klassendiagramme bestehend aus Klassen, sowie deren Attribute und Referenzen, verwendet.
- oAW verwendet dazu EMF (eclipse modeling framework)
- Eigene Datentypen können in EMF durch Aufzählungen definiert werden
- Bei Referenzen wird zwischen „normalen Referenzen“ und Komposition unterschieden

Definition of Underlying Meta-Model: UML-Model



Konsistenzprüfung

- Vor der Codegenerierung ist eine Prüfung der Korrektheit der Modelle möglich
- oAW unterstützt die Definition von Testbedingungen in Prädikatenlogik 1. Stufe
- Beispiel:

```
context Actor ERROR "Actor "+ this+": Name must be set and be unique":
    (this.name!=null) &&
    (eRootContainer.eAllContents.typeSelect(Actor).select(a | a.name== this.name).size==1);
```

- Erläuterung:
  - this: referenziert das aktuelle Objekt
  - eRootContainer: referenziert das Wurzelelement
  - eAllContents: liefert eine Liste aller vom aktuellen Objekt enthaltenen Objekte
  - typeSelect: beschränkt eine Menge auf alle Objekte des angegebenen Typs
  - select: wählt alle Element aus einer Menge, die die angegebene Bedingung erfüllen