Automatisierte Architekturanalyse unter Einsatz von UML 2.0 Modellen

Vortrag AW1

Allgemeines zum Vortrag

- Vorstellung: Thorben Pergande
- Bisheriges Studium: B.Sc. Angewandte Informatik an der HAW
- Professoren an dieser Ausarbeitung beteiligt:
  - Prof. Zukunft (Pervasive Gaming)
  - Prof. Sarstedt (MI)
- Fragen sind jederzeit erwünscht!

Agenda

- Motivation
- Problemstellung
- Lösungsansatz
- Risiken
- Ausblick AW2

Motivation

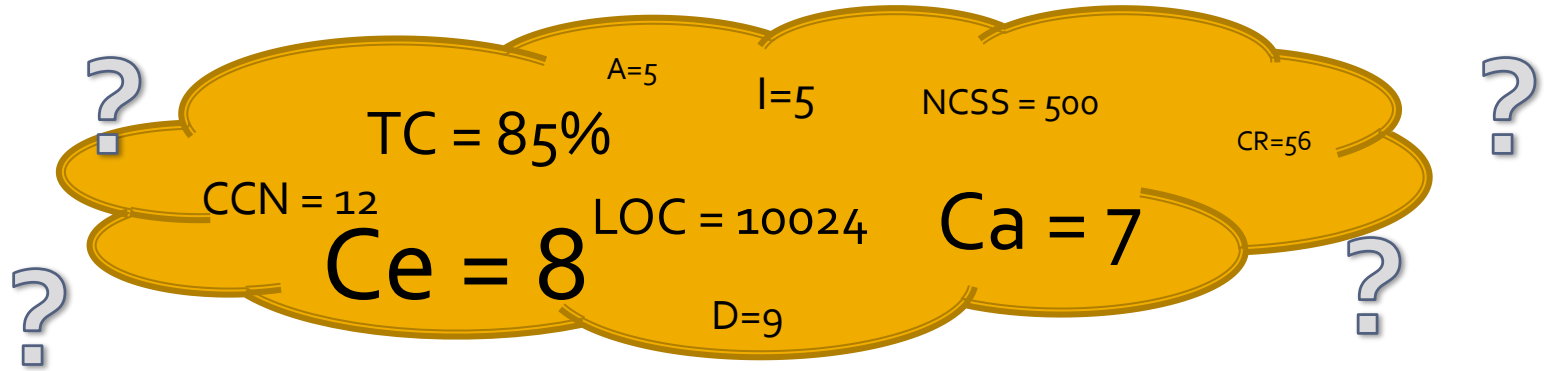
- Statische Architekturanalyse:
  Untersuchung der inneren Struktur eines Softwaresystem um dieses zu Verstehen und zu Bewerten.

- **Ziel**: mögliche Gefahren für die Qualität der Software finden

Motivation

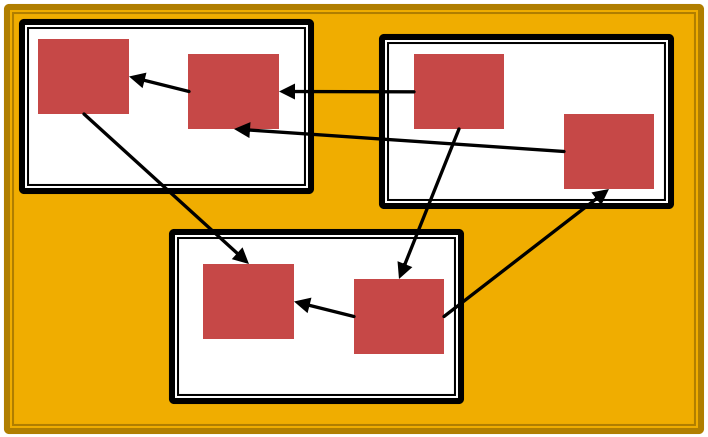
- Unterscheidung zwischen:
  - SOLL-Architektur: Modelle, Beschreibungen
  - IST-Architektur: Quelltext
- **Bisher**: Architekturanalyse meist metrikenbasiert

  → **Messwerte und viele Zahlen**



Problemstellung

- Die Architektur eines Systems degeneriert



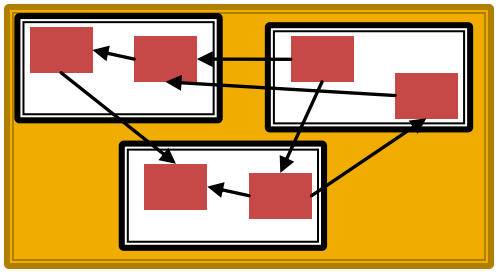
```
//generated Stub (SOLL)
public long calc (EK price, Marge percent)
{
//to be filled with logic
}
```

!!!

```
//generated Stub (SOLL)
public long calc (EK price, Marge percent)
{
  long res= price.getEK() *percent.asLong()
  Rabatt rab = facility.get(„Weihnachtsrabatt“);
  return res * rab.asLong();
}
```

Problemstellung

- Architekturanalyse (statisch) um Degeneration zu erkennen zu minimieren
- UML 2.0 Modelle als Darstellung der SOLL-Architektur
- Wie kann die SOLL-Architektur mit der IST-Architektur verglichen werden?




```
//generated Stub (SOLL)
public long calc (EK price, Marge percent)
{
  long res= price.getEK() *percent.asLong()
  Rabatt rab = facility.get(„Weihnachtsrabatt“);
  return res* rab.asLong();
}
```

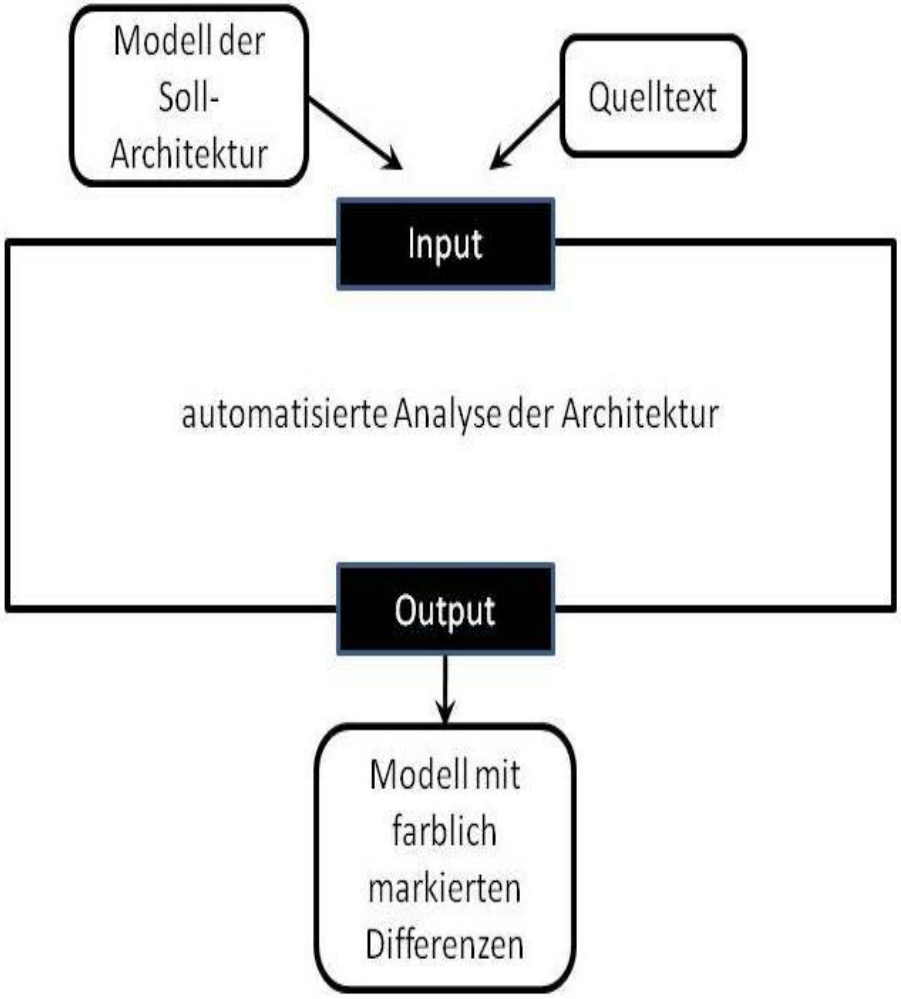
Vision

- **Ziel**: Modelle als SOLL-Architektur direkt mit Quelltext als IST-Architektur vergleichen und die Ergebnisse geeignet darstellen

- **Ergebnis**: Modell mit Markierungen bei:
  - Artefakte / Beziehungen **zu wenig**
  - Artefakte / Beziehungen **zu viel**

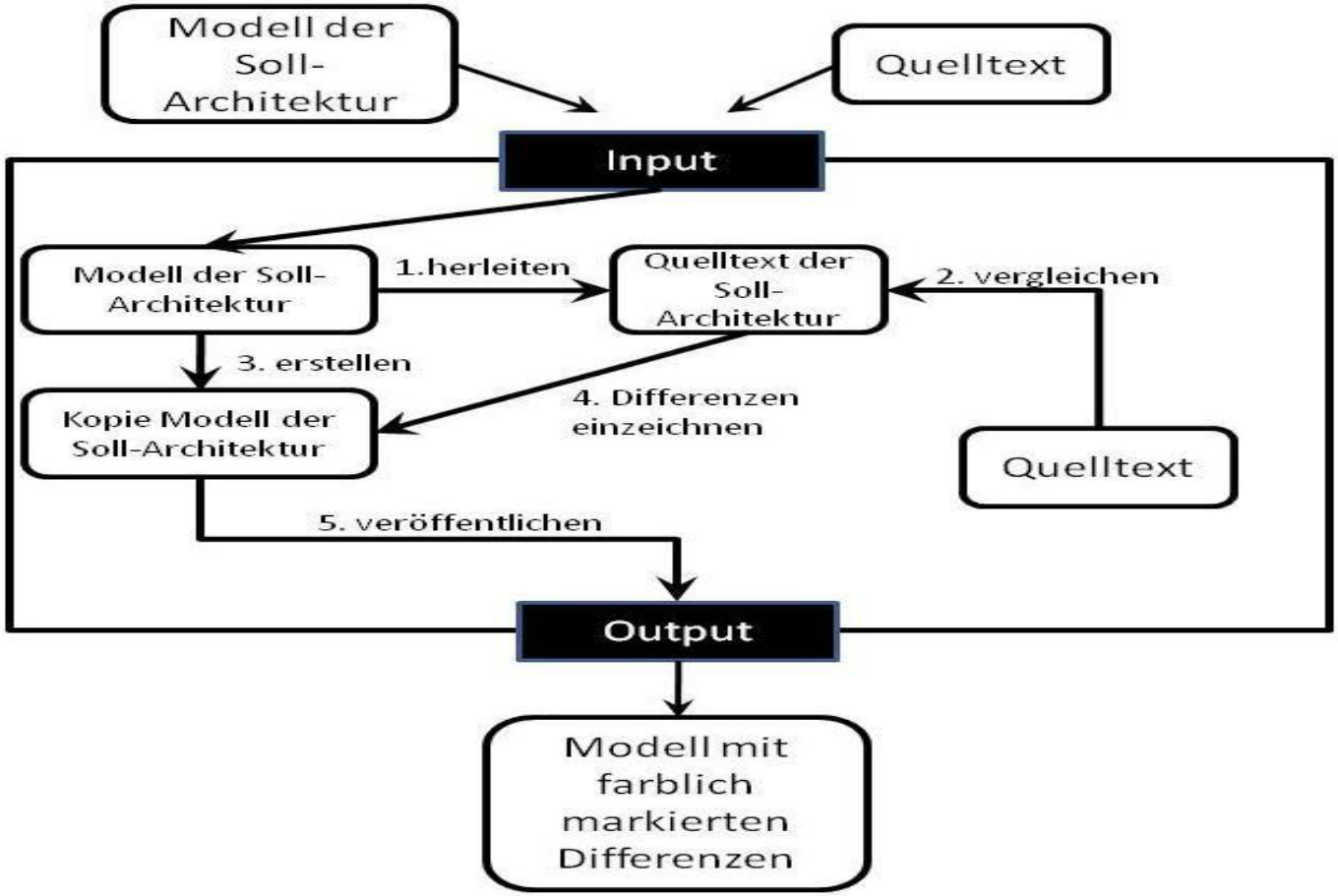
Vision

- **Black-Box**: Ergebnis der Analyse einfacher zu interpretieren, da als **bekanntes** Modell vorhanden



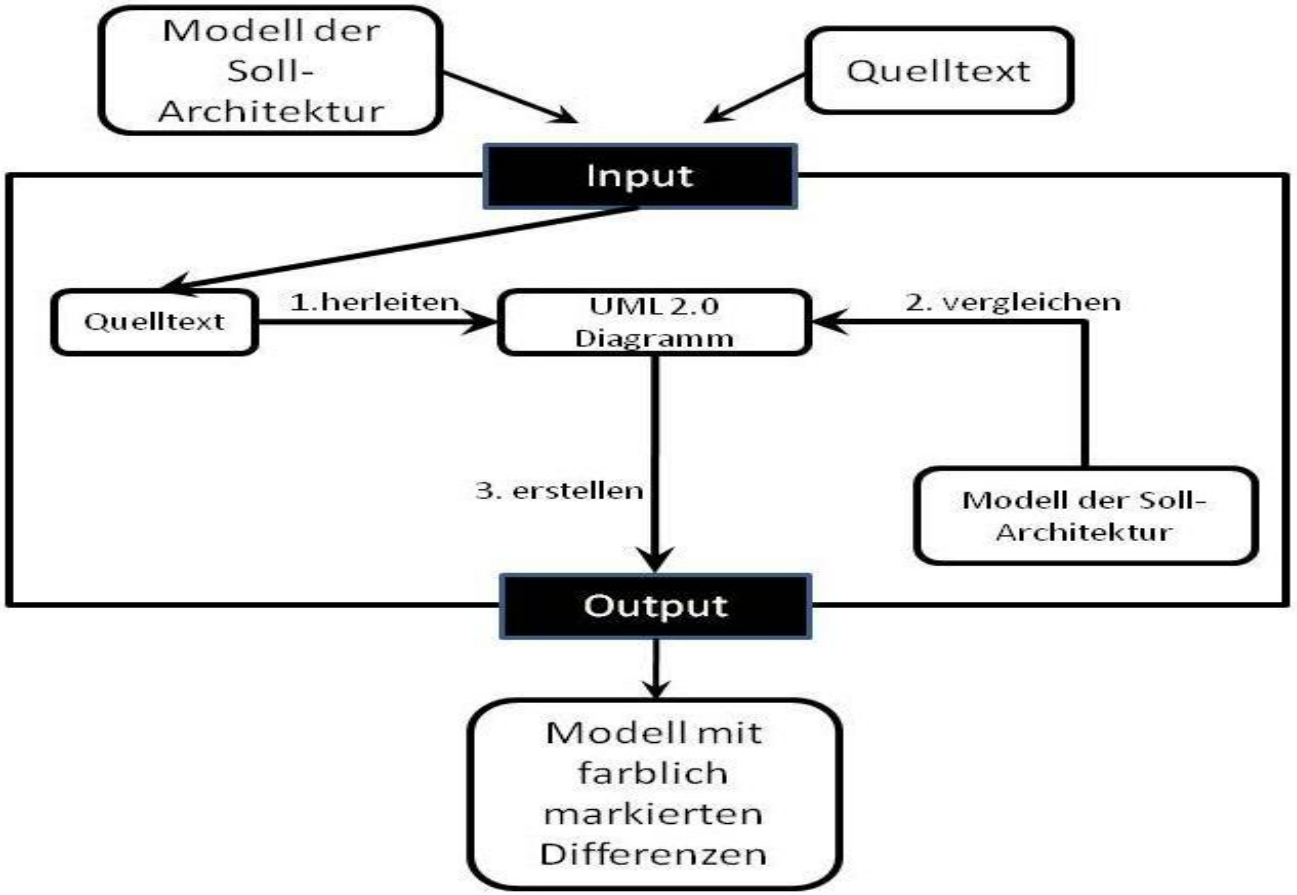
Lösungsansätze

- Top-Down Analyse



Lösungsansätze

- Bottom-Up Analyse

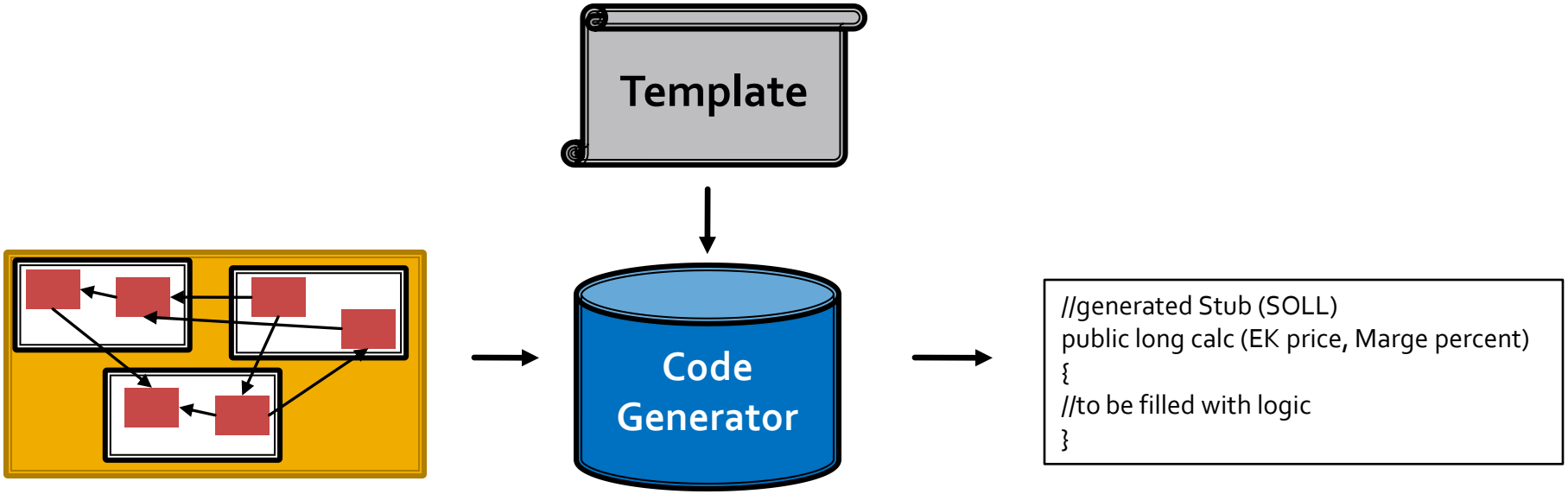


Techniken

- Codegenerierung nach MDSD
  - Modell einlesen:
    - Identifizierung welcher Modelltyp
    - Sammeln der Artefakte (Komponenten, Beziehungen…)
    - Übersetzung in Metamodell (z.B. MOF)→
  - Übersetzen Metamodell in Zielsprache
    - Semantik definieren, z.B. über Templates
    - Codegeneratoren vorhanden, Templates erstellen aber aufwendig!

Techniken

- Codegenerierung nach MDSD
  - Unterteilung in generierten und manuellen Code!



Techniken

- Modellgenerierung
  - aus dem Programmfluss heraus sollen Modelle erstellt oder verändert werden
  - UML 2.0 IDE mit solch einer Schnittstelle nötig
  - Bsp.: Visio und primäre Interopassembly

Techniken

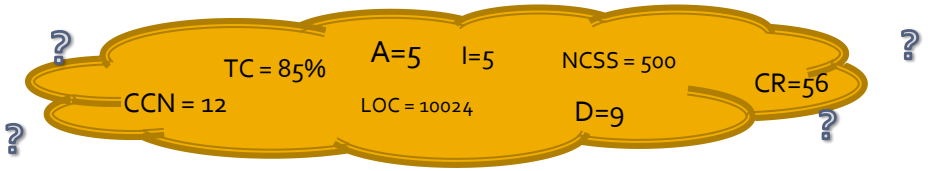
- Vergleich von Quelltexten
  - Quelltext ist in zwei Teilen unterschieden
    - Generierte Teile
    - Manuelle Erweiterungen (fachliche Logik)
  - Differenzen bzgl. **Artefakte & Beziehungen** müssen gefunden werden
  - Recherche nach einem geeigneten Werkzeug läuft noch

Risiken

- Komplexität der einzelnen Module hoch
  - Einarbeitungszeit, Anpassungen
- Bottom-Up Ansatz benötigt eine Modellgeneration aus Quelltext heraus
- Top-Down Ansatz benötigt Quelltextvergleich
  - Wenn kein geeignetes Werkzeug gefunden wird → **Eigenentwicklung**

Risiken

- Bisher SOLL-Architektur nicht analysiert, dieses sollte auch geschehen!
  - Weiteres Modul, dass die SOLL-Architektur prüft, z.B. metrikenbasiert
  - Weiteres Modul mit hoher Komplexität/Zeitaufwand
  - Somit auch Performance-Analyse durchführbar, aber geeignete Repräsentation der Ergebnisse schwierig



Ausblick AW2

- Prototyp erstellen für:
  - **Eine** Programmiersprache
  - Einen Diagrammtypen, z.B. **Komponentendiagramm**
  - Vermutlich **Top-Down** Analyse, da Risiko geringer, aber Anzahl der Schritte pro Analyse höher
  - Möglichst viele vorhandene Komponenten versuchen zu **bündeln** → modularer Aufbau
- Wahrscheinlich .Net-Umgebung da:
  - Interaktion mit MS Visio möglich und dokumentiert
  - Codegeneratoren-Basis vorhanden

Fragen

- …und hoffentlich Antworten…