Model Correspondences in Continuous Engineering of Mediator-based Information Systems

Overview

- Mediator-based Information Systems (MBIS)
- MoCA Language
- MoCAs in Continuous Engineering of MBIS

Mediator-based Information System (MBIS)

Correspondences

- Intensional Correspondences
- Extensional Overlap
- Object fusion

Specification Language for Model Correspondence Assertions (MoCA)

Objective: Conceptual Design of MBIS

Analysis
- Functional Requirements
- Autonomy

Design
- Independence of Implementation
- Formal Definition

Impl. GaV, LaV, QCA
Principle: ‘Separation of Concerns’ (1)

Universe of Discourse

Model Extension State of the Info. System

\[ \subseteq \]

Real World State

Intensional Model Level

Extensional Model Level

Real World

MoCA Language – Overview

MoCA Language – Example

component overlap MoCA-Example

Schema1 ~ Schema2

import ...

correspondence Person

elements Person = Person

extension

Schem1.Person ~ Schem2.Person

with ID = ID identifying

intension

ID = ID

Name = (Name, Title) with ...

Works = Works

end correspondence Company ...

end

end component overlap
MoCA Language – Formal Semantics (1)

MoCA Language – Formal Semantics (2)

MoCA Language – Summary

Different Kinds of MBIS and model correspondences
→ Different Semantics of the extensional correspondence relationship

Functional Requirements
→ 'Separating of concerns'
→ Intensional and extensional part of a MoCA specification
→ Object Fusion NOT inside a MoCA

Autonomy
→ mostly 1:1 model correspondences

Independence of Implementation, Conceptual Design
→ bi-directional mappings
→ oo data model
→ no transformation of schemas to resolve structural heterogeneity

Definition of Syntax and Semantics
→ Formalization on the basis of algebraic specifications

MoCAs in Continuous Engineering of MBIS (1):
Design of a Mediator Schema
MoCAs in Continuous Engineering of MBIS (2): Implementation of the Mediator

MoCAs in Continuous Engineering of MBIS (1): Evolution of MBIS

Continuous Software Engineering (CSE)

Evolution: Propagation of Modifications
Classification of Evolution Operations (1)

PrimitiveEvolution

ModelEvolution

CIS

Classfi

Classification of Evolution Operations (2)

Evolution (Actions)

PrimitiveEvolution

CompositeEvolution

FACHBEREICH 13

INFORMATIK

Computergestützte

Informationssysteme CIS

Classification of Evolution Operations (3):
Composite ModelEvolution

1. SplitRecord

2. AttributeFusion

3. SeparateAttributes

4. IncludeAttributes

5. MoveAttributes

Classification of Evolution Operations (3):
Composite ModelEvolution

1. SplitRecord

2. AttributeFusion

3. SeparateAttributes

4. IncludeAttributes

5. MoveAttributes

Specification of Evolution Operations: Metamodelling Approach

Metamodel Level

Intensional MoCA Elements

+Definition

| Correspondence |

| Concept Element |

| +Type |

| Extensional Model Elements |

| Concept Element Instance |

Model Level

MoCAs

Use

Schemas

Person

Person

+Type

Data

«instance of»

«instance of»

«instance of»
**Formalization of Evolution Operations (1)**

**Action act**

**Precondition** τ

**Effect** γ

**Propagation** δ

This represents the following dynamic formula:

\((\{\text{act}\} \text{true} \rightarrow \text{τ}) \land (\{\text{act}\} \gamma \land \delta)\)

---

**Formalization of Evolution Operations (2)**

**Action delClass(m: Model, c: Class)**

**Precondition**
The model contains the class.

\(c \in \text{modelElement}(m)\)

**Postcondition**
The class does not exist anymore.

\(\neg c \in \text{modelElement}(m)\)

**Propagation**

Model level: All features, operations and relationships of the class do not exist.

\((\forall \text{a: Attribute} \cdot \text{a} \in \text{feature}(c) \rightarrow \text{delAttribute}(c, \text{a}) \land \text{true})\)

This represents the following dynamic formula:

\((\forall \text{op: Operation} \cdot \text{op} \in \text{op} \rightarrow \text{delOperation}(c, \text{op}) \land \text{true})\)

**MoCA level:** All concept elements that are based on the class do not exist.

\((\forall \text{ce} \in \text{conceptElement}(c) \cdot \text{delConceptElement}(ce) \land \text{true})\)
**Formalization with Graph Transformation**

\[ \text{delClass} \quad (m: \text{Model}, c: \text{Class}) \]

\[ a) \quad \text{m: Model} \quad \text{ModelElement} \quad \text{\rightarrow} \quad \text{m: Model} \]

\[ \land \forall a \in \text{feature(c)} \cdot \text{deleteAttribute(c,a)} \]

\[ \land \ldots \]

\[ \land \text{ce} \in \text{ConceptElement} \ldots \]

\[ b) \quad \text{m: Model} \quad \text{ModelElement} \quad \text{\rightarrow} \quad \text{m: Model} \]

**Summary**

- Mediator-based Information Systems provide a read access to tightly-coupled autonomous data sources
- Schemas and correspondences are constituent parts of MBIS
- MoCA Language for Conceptual Design of MBIS, independent of implementation
- MoCA Language has to be embedded into continuous engineering of MBIS
- Schema evolution requires the propagation of modifications to MoCAs; schema modification is the most interesting evolution operation in MBIS
- Metamodeling is suitable to formalize evolution operations