Model Consistency ensured by Metamodel Integration
MoConseMI
6th GEMOC 2018, Copenhagen

Johannes Meier      Andreas Winter

Software Engineering Group
Department of Computing Science
Carl von Ossietzky University, Oldenburg, Germany

15. October 2018
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content

- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content

- Ensure Consistency between Artifacts automatically

<table>
<thead>
<tr>
<th>ID</th>
<th>Author</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Andreas Winter</td>
<td>The student must be able to register for an event.</td>
</tr>
<tr>
<td>R2</td>
<td>Johannes Meier</td>
<td>The student must be enrolled at the university.</td>
</tr>
</tbody>
</table>
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content

- Ensure Consistency between Artifacts automatically

<table>
<thead>
<tr>
<th>ID</th>
<th>Author</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Andreas Winter</td>
<td>The student must be able to register for an event.</td>
</tr>
<tr>
<td>R2</td>
<td>Johannes Meier</td>
<td>The student must be enrolled at the university.</td>
</tr>
</tbody>
</table>

University university[1] students[*]

Student
  +Name : String
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, ...
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, ...
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically

<table>
<thead>
<tr>
<th>ID</th>
<th>Author</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Andreas Winter</td>
<td>The student must be able to register for an event.</td>
</tr>
<tr>
<td>R2</td>
<td>Johannes Meier</td>
<td>The student must be enrolled at the university.</td>
</tr>
</tbody>
</table>
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, ...
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, ...
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, ...
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Motivation

- various Artifacts in Software Development:
  - Diagrams, DSLs, Tools, …
  - Artifacts are technically separated
  - Artifacts are interrelated regarding content
- Ensure Consistency between Artifacts automatically
Problem

There are further Software Development Projects:
- e.g. with formal Specifications, C++, Test Cases, Documentation, Project Management, Build Tools
- Traceability
- $\rightarrow$ further Consistency issues

General Problem:
- Artifacts are technically separated, but interrelated contentwise
- specific Consistency Rules have to be fulfilled automatically

Goal

Ensure Consistency between Artifacts automatically!
- Artifact == Model + Metamodel (structural formalization [CNS12])
- $\rightarrow$ Model Integration
Challenges
Challenges

1. Formalize Consistency Rules
Challenges

1. Formalize Consistency Rules
Challenges

1. Formalize Consistency Rules

Each Requirement is linked with all Methods whose Name is contained in the Text of the Requirement.

Sourcecode and Class Diagrams describe the same set of classes, identified by their class name.
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)

- reuse Model Techniques which work only with one Model
- used as single Point of Truth
- Single Underlying Model [ASB09]
- SUMM and SUM are explicit
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)

- reuse Model Techniques which work only with one Model
- used as single Point of Truth
- Single Underlying Model [ASB09]
- SUMM and SUM are explicit
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models

- existing Metamodels: DSLs, Environments, Tools, …
- existing Models: ongoing projects, legacy data, …
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models

- existing Metamodels: DSLs, Environments, Tools, ...
- existing Models: ongoing projects, legacy data, ...

<table>
<thead>
<tr>
<th>ClassDiagram</th>
<th>JavaASG</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagram [0..1]</td>
<td>asg [0..1]</td>
</tr>
<tr>
<td>class [1]</td>
<td>calledBy [+]</td>
</tr>
<tr>
<td>associations [+]</td>
<td>calling [+]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClassType</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>className : EString [1]</td>
<td>name : EString [1]</td>
</tr>
<tr>
<td>classes [+]</td>
<td>methods [+]</td>
</tr>
<tr>
<td>class [1]</td>
<td>calling [+]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>content [+]</td>
</tr>
<tr>
<td>container [1]</td>
</tr>
<tr>
<td>id : EString [0..1]</td>
</tr>
<tr>
<td>author : EString [0..1]</td>
</tr>
<tr>
<td>text : EString [0..1]</td>
</tr>
</tbody>
</table>

Motivation
Challenges

1. Formalize Consistency Rules

2. Create explicit SUM(M)

3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
Challenges

1. Formalize Consistency Rules

2. Create explicit SUM(M)

3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models

4. Ensure Model Consistency
Related Work

<table>
<thead>
<tr>
<th></th>
<th>synthetic</th>
<th>OSM</th>
<th>projectional</th>
<th>Vitruvius</th>
<th>GEMOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Formalize Consistency Rules</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2. Create explicit SUM(M)</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>3a. Reuse initial Models</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>3b. Fix initial Inconsistencies</td>
<td>✓</td>
<td>–</td>
<td>✗</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>3c. Consistent initial Models</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4. Ensure Model Consistency</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- ISO Standard 42010:2011 [IEE11]: synthetic vs. projectional
- synthetic: TGGs [SK08], QVT-R [RJV09], explicit correspondences [EEC+14]
- OSM: Single Underlying (Meta)Model (SUM(M)) [ASB09]
- Vitruvius [KBL13, BHK+14]
- GEMOC Approach [LDC18]
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

Activities
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

Activities

1. Configuration of Operators

SUMM
Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

Activities

1. Configuration of Operators

SUM

Requirements

Java

ClassDiagram

Add Association

Change Multiplicity

Merge Classes

SUM(M)

Change Multiplicity

Merge Attributes

Add Association

Change Multiplicity

Merge Classes

SUM(M)
Metamodel Integration

Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent inital Models
4. Ensure Model Consistency

Activities

1. Configuration of Operators
2. Initialization of SUM
3. Consistency Assurance

Java
ClassDiagram

Requirements 1 Add Association 2 Change Multiplicity 3 Change Multiplicity 4 Merge Classes 5 Merge Attributes 6 SUM(M)
Metamodel Integration

**Challenges**

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

**Activities**

1. Configuration of Operators
2. Initialization of SUM

**Requirements**

- Java
- ClassDiagram

1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)
**Metamodel Integration**

**Challenges**

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

**Activities**

1. Configuration of Operators
2. Initialization of SUM
3. Consistency Assurance

---

**Diagram:**

- **Requirements**
  - 1. Delete Association
  - 2. Change Multiplicity
  - 3. Change Multiplicity
  - 4. Split Class
  - 5. Split Attribute
  - 6. SUM(M)

- **Tools:**
  - Java
  - ClassDiagram
Metamodel Integration

**Challenges**

1. Formalize Consistency Rules

2. Create explicit SUM(M)

3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models

4. Ensure Model Consistency

**Activities**

1. Configuration of Operators

2. Initialization of SUM

3. Consistency Assurance

---

**Requirements**

1. Add Association

2. Change Multiplicity

3. Change Multiplicity

4. Merge Classes

5. Merge Attributes

6. SUM(M)
Metamodel Integration

Challenges

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   a. Reuse initial Models
   b. Fix initial Inconsistencies
   c. Consistent initial Models
4. Ensure Model Consistency

Activities

1. Configuration of Operators
2. Initialization of SUM
3. Consistency Assurance

Requirements

1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)

Java
ClassDiagram

1x Methodologist

Johannes Meier, Andreas Winter
Metamodel Integration

**Challenges**

1. Formalize Consistency Rules
2. Create explicit SUM(M)
3. Support initial (Meta)Models:
   - a. Reuse initial Models
   - b. Fix initial Inconsistencies
   - c. Consistent initial Models
4. Ensure Model Consistency

**Activities**

1. Configuration of Operators
2. Initialization of SUM
3. Consistency Assurance

**Requirements**
- Java
- ClassDiagram

**SUM(M)**

**Add Association**

1. **Change Multiplicity**
2. **Change Multiplicity**
3. **Merge Classes**
4. **Merge Attributes**

Johannes Meier, Andreas Winter

15. October 2018
1. Configuration of Operators
1. Configuration of Operators

Metamodel Integration

Chechik, Nejati, Sabetzadeh: A Relationship-Based Approach to Model Integration (2012)


Requirements

Java

ClassDiagram

1. Add Association

2. Change Multiplicity

3. Change Multiplicity

4. Merge Classes

5. Merge Attributes

6. SUM(M)
1. Configuration of Operators

**Metamodel Integration**

- **1. Configuration of Operators**

  - **Metamodel Decisions**
  - **Model Decisions**

  - **Operator**

  - **MM** → **Δ_{MM}** → **MM'**
  - **M** → **Δ_{M}** → **M'**

- **Requirements**

  - **Java**
  - **ClassDiagram**

  - **1. Add Association**
  - **2. Change Multiplicity**
  - **3. Change Multiplicity**
  - **4. Merge Classes**
  - **5. Merge Attributes**
  - **6. SUM(M)**
1. Configuration of Operators

- **MM** → **MM'**
- **M** → **M'**

**Operator**

- **Δ_{MM}**
- **Δ_{M}**

**Metamodel**

- **Decisions**

**Model**

- **Decisions**

**Requirements**

1. **Add Association**
2. **Change Multiplicity**
3. **Change Multiplicity**
4. **Merge Classes**
5. **Merge Attributes**
6. **SUM(M)**

- **Java**
- **ClassDiagram**

**References**

- Chechik, Nejati, Sabetzadeh: *A Relationship-Based Approach to Model Integration* (2012)

→ merge, composition, weaving

→ extended Coupled Operators
1. Configuration of Operators

Metamodel Integration

Chechik, Nejati, Sabetzadeh: A Relationship-Based Approach to Model Integration (2012)

merge, composition, weaving


extended Coupled Operators

Johannes Meier, Andreas Winter

Metamodel Integration

15. October 2018 7 / 12
1. Configuration of Operators

- **Operator**
- **Inverse Operator**

Metamodel Decisions

Model Decisions

### Requirements

1. Java
2. Add Association
3. Change Multiplicity
4. ClassDiagram
5. Change Multiplicity
6. Merge Classes

SUM(M)

merge, composition, weaving

Chechik, Nejati, Sabetzadeh: A Relationship-Based Approach to Model Integration (2012)

1. Configuration of Operators

[Diagram showing the configuration of operators between metamodels MM and MM', models M and M', and their respective decision-making processes.]

- **Metamodel Integration**
- **Operator**
- **Inverse Operator**
- **Model Decisions**
- **Metamodel Decisions**

- **Requirements**
  - Java
  - Add Association
  - Change Multiplicity
  - Merge Classes
  - Merge Attributes
  - SUM(M)

- **References**
  - Chechik, Nejati, Sabetzadeh: A Relationship-Based Approach to Model Integration (2012)
2. Initialization of SUM: Overview
2. Initialization of SUM: Overview

Java

ClassDiagram

Requirements

1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. Merge Attributes

SUM(M)

RequirementsSpecification

container [1]
content [•]

Requirement

rowNumber : EInt [0..1]
id : EString [0..1]
author : EString [0..1]
text : EString [0..1]

R1 : Requirement

rowNumber : EInt [0..1] = 1
id : EString [0..1] = "R1"
author : EString [0..1] = "Andreas Winter"
text : EString [0..1] = "The student must be able to register for an event."

R2 : Requirement

rowNumber : EInt [0..1] = 2
id : EString [0..1] = "R2"
author : EString [0..1] = "Johannes Meier"
text : EString [0..1] = "The student must be enrolled at the university."
2. Initialization of SUM: Overview

Requirements

1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)
2. Initialization of SUM: Overview

- **Initialization of SUM: Overview**

  **Requirements**
  1. **Java**
  2. **Add Association**
  3. **Change Multiplicity**
  4. **Change Multiplicity**
  5. **Merge Classes**
  6. **Merge Attributes**

  **SUM(M)**
2. Initialization of SUM: Overview

- **Requirements**:
  - Add Relation
  - Change Multiplicity
  - Change Multiplicity
  - Merge Classes
  - Merge Attributes

- **Java**:
  - Requirements
  - ClassDiagram

- **ClassType**:
  - name : EString [1]

- **Method**:
  - name : EString [1]

- **ProjectData**:
  - integrator [0..1]

- **JavaASG**:
  - asg [1]

- **RequirementsSpecification**:
  - container [1]
  - content [+]

- **Requirement**:
  - rowNumber : EInt [0..1]
  - id : EString [0..1]
  - author : EString [0..1]
  - text : EString [0..1]

- **ClassDiagram**:
  - Container 1
  - Content 1

**Requirements Specification**

1. Initialization of SUM: Overview
2. Requirements
3. JavaASG
4. ProjectData
5. RequirementsSpecification
6. Requirement

Johannes Meier, Andreas Winter

Metamodel Integration

15. October 2018
2. Initialization of SUM: Overview

Requirements

1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)
2. Initialization of SUM: Overview

**Requirements**
1. **Add Association**
2. **Change Multiplicity**
3. **Change Multiplicity**
4. **Merge Classes**
5. **Merge Attributes**
6. **SUM(M)**
2. Initialization of SUM: Overview

Requirements
1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)

Metamodel Integration
15. October 2018
2. Initialization of SUM: Overview

**Requirements**

1. Java
2. Add Association
3. Change Multiplicity
4. Change Multiplicity
5. Merge Classes
6. Merge Attributes

**Java**

**ClassDiagram**

**Requirements**

**SUM(M)**
2. Initialization of SUM: Overview

**Requirements**
1. Java
2. Add Association
3. Change Multiplicity
4. Change Multiplicity
5. Merge Classes
6. Merge Attributes

**SUM(M)**
2. Initialization of SUM: Overview

- **Add Association**
  - Requirement R1: The student must be able to register for an event.
  - Requirement R2: The student must be enrolled at the university.

- **Change Multiplicity**
  - M1: Method name = "register"
  - M2: Method name = "start"

- **Merge Attributes**
  - Class University
  - Class Student

- **SUM(M)**
2. Initialization of SUM: Details
2. Initialization of SUM: Details
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models inverse to each other

Requirements

1. Add Association
2. ChangeMultiplicity
3. ChangeMultiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)

Java

ClassDiagram

ClassType

name : String

Class

className : String

: ClassType

name = “University”

className = “University”

: ClassType

name = “Student”

className = “Student”

Metamodel Changes

Model Changes
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models inverse to each other
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models inverse to each other

Requirements

1. Add Association

2. Change Multiplicity

3. Change Multiplicity

4. Merge Classes

5. Merge Attributes

SUM(M)

Java

ClassDiagram

SUM(M)
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models

SUM(M)

ClassType
name : String
className : String

Class
name = "University"
className = "University"

Class
name = "Student"
className = "Student"

Class
name = "University"
className = "University"

Class
name = "Student"
className = "Student"

Java

ClassType
name : String

Class
name = "University"
className = "University"

Class
name = "Student"

ClassDiagram

Metamodel Changes

Model Changes

Model Changes

Metamodel Changes

Requirements

1. Java
2. Add Association
3. ClassDiagram
4. Change Multiplicity
5. Split Class
6. Merge Attributes

Johannes Meier, Andreas Winter

Metamodel Integration

15. October 2018
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models

SUM(M)

ClassType
name : String

Class
className : String

ClassType
name = “University”
className = “University”

ClassType
name = “Student”

Class
name = “University”
className = “Student”

Class
name = “Student”
className =

ClassType
name : String

Class
className : String

Split Class

Merge Classes

inverse to each other

JAVA

ClassDiagram

Model

Changes

Model

Changes

Metamodel

Changes

Requirements

Add Association

Change Multiplicity

Split Class

SUM(M)

Model

Changes

Metamodel

Changes

Model

Changes

Metamodel

Changes

Johannes Meier, Andreas Winter

Metamodel Integration

15. October 2018 9 / 12
2. Initialization of SUM: Details

3a. Forward Execution to create the initial SUM

3b. Backward Execution to fix the initial Models

- **Java**
  - ClassType
    - name: String

- **ClassDiagram**
  - Class
    - className: String

- **SUM(M)**
  - ClassType
    - name: String
    - className: String

### Merge Classes

- ClassType: name = "University"
  - className = "University"

- ClassType: name = "Student"
  - className = "Student"

### Split Class

- ClassType: name = "University"
  - className = "University"

- ClassType: name = "Student"
  - className = "Student"

### Model Changes

- Add Association
- Change Multiplicity
- Split Class
3. Consistency Assurance: Details
3. Consistency Assurance: Details

```
1. Initialization (forward)
2. Run (backward)
3. Run (forward)
```

```
1. Create Link between R2 "enrole" and M2 "start"
2. Rename Method M1 from "register" to "enrole"
3. inverse to each other
```

```
1. Add Association
2. Change Multiplicity
3. Change Multiplicity
4. Merge Classes
5. Merge Attributes
6. SUM(M)
```
3. Consistency Assurance: Details

1. Initialization (forward)

   1. Create Link between R2 "enrole" and M2 "start"
   2. Rename Method M1 from "register" to "enrole"

2. Run (backward)

   1. Fulfill R1 "register"
   2. Fulfill R2 "enrole"

3. Run (forward)

   1. Fulfill R1 "register"
   2. Fulfill R2 "enrole"
3. Consistency Assurance: Details

1. Initialization (forward)

   1. Create Link between R2 "enrole" and M2 "start"
   2. Rename Method M1 from "register" to "enrole"

   1. Add Association
   2. Change Multiplicity
   3. Change Multiplicity
   4. Merge Classes
   5. Merge Attributes
   6. SUM(M)
3. Consistency Assurance: Details

1. Initialization (forward)
   - Create Link between R2 “enrole” and M2 “start”
   - Rename Method M1 from “register” to “enrole”

2. Run (backward)
   - 1. Delete Association
   - 2. Change Multiplicity
   - 3. Change Multiplicity
   - 4. Split Class Into Two
   - 5. Split Attribute
   - 6. Add Association
3. Consistency Assurance: Details

1. Initialization (forward)
   - Create Link between R2 "enrole" and M2 "start"
   - Rename Method M1 from "register" to "enrole"

2. Run (backward)
   - Delete Association

Requirements

Add Association

Delete Association

Java

ClassDiagram

SUM(M)

1. Initialization (forward)
   - Create Link between R2 "enrole" and M2 "start"
   - Rename Method M1 from "register" to "enrole"

2. Run (backward)
   - Delete Association

Java

ClassDiagram

SUM(M)
3. Consistency Assurance: Details

1. Initialization (forward)
   - Create Link between R2 "enrole" and M2 "start"
   - Rename Method M1 from "register" to "enrole"

2. Run (backward)
   - Delete Association (forward)
   - Requirement R1: "... register ..."
   - Method M1: "register"
   - Requirement R2: "... enrole ...
   - Method M2: "start"

Add Association

Delete Association

Model Changes

Metamodel Changes

Requirements

Java

SUM(M)

ClassDiagram

SUM(M)
3. Consistency Assurance: Details

1. Initialization (forward)

2. Run (backward)

3. Run (forward)

Add Association:
- R1 : Requirement
text = "... register ...
- M1 : Method
name = "register"
- R2 : Requirement
text = "... enrole ...
- M2 : Method
name = "start"

Delete Association:
- R1 : Requirement
text = "... enrole ...
- M2 : Method
name = "start"

User:
- 1. Create Link between R2 "enrole" and M2 "start"
- 2. Rename Method M1 from "register" to "enrole"

SUM(M):
- R1 : Requirement
  text = "... register ...
  fulfilled
  fulfilledBy
- M1 : Method
  name = "register"
- R2 : Requirement
  text = "... enrole ...
  fulfilled
  fulfilledBy
- M2 : Method
  name = "start"

Requirements
- Requirement
  text : String
- Method
  name : String

Java
- Requirement
  text : String
- Method
  name : String

Metamodel Changes
- Requirements
- Java

Johannes Meier, Andreas Winter
Metamodel Integration
15. October 2018 10 / 12
3. Consistency Assurance: Details

1. Initialization (forward)
2. Run (backward)
3. Run (forward)

Add Association

Delete Association

Add Association

User

Johannes Meier, Andreas Winter
3. Consistency Assurance: Details

1. Initialization (forward)
2. Run (backward)
3. Run (forward)

- **Requirements**
  - **R1**: Requirement
    - text: "... register ...
  - **R2**: Requirement
    - text: "... enrol ...

- **Java**
  - **M1**: Method
    - name: "register"
  - **M2**: Method
    - name: "start"

- **SUM(M)**
  - **R1**: Requirement
    - text: "... register ...
  - **R2**: Requirement
    - text: "... enrol ...

**Add Association**

1. Create Link between R2 "enrole" and M2 "start"
2. Rename Method M1 from "register" to "enrole"

**Delete Association**

**Add Association**

**User**

**Model Changes**

**Metamodel Changes**

Johannes Meier, Andreas Winter
3. Consistency Assurance: Details

1. Initialization (forward)
   - Create Link between R2 “enrole” and M2 “start”
   - Rename Method M1 from “register” to “enrole”

2. Run (backward)

3. Run (forward)

Requirements
- **R1**: Requirement
  - text: “... register ...”
- **R2**: Requirement
  - text: “... enrol ...”

Java
- **M1**: Method
  - name: “register”
- **M2**: Method
  - name: “start”

SUM(M)
- **R1**: Requirement
  - text: “... register ...”
  - fulfilled
  - fulfilledBy
- **R2**: Requirement
  - text: “... enrol ...”
  - fulfilled
  - fulfilledBy

Add Association

Delete Association

Add Association

User

Model Changes

Johannes Meier, Andreas Winter

Metamodel Integration

15. October 2018
3. Consistency Assurance: Details

Add Association

1. Initialization (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

2. Run (backward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

3. Run (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

Delete Association

1. Run (backward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

2. Run (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

Add Association

1. Run (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

2. Run (backward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

Model Changes

1. Initialization (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

2. Run (backward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

3. Run (forward)
   - R1: Requirement
     - text = "... register ...
   - M1: Method
     - name = "register"

Delta User

1. Create Link between R2 "enrole" and M2 "start"
2. Rename Method M1 from "register" to "enrole"
Operators: Summary

- **Metamodel Change** $\Delta_{MM}$:
  small Change in the Metamodel

- **Model Change** $\Delta_{M}$:
  handle Model-Co-Evolution (Coupled Operators [HWW11])

- Configurations by the Methodologist:
  - **Metamodel Decisions**: set Properties for wanted Metamodel Changes
  - **Model Decisions**: describe Model Changes for Consistency Rules

- **Bi-Directionality only for MM**: combine with inverse Operator

- currently 20 Operators implemented
Summary

Operator-based bottom-up SUM-Approach for Model Consistency:

- Methodologist configures arbitrary, but stable Chain of configured Operators (once)
- User applies Changes and Model Consistency is ensured automatically by executing the Operator Chain
- → separated Models are migrated to projectional Views on the SUM
Conclusion

Literature I


Conclusion

Literature II


