Co-evolution of Metamodels and Models through Consistent Change Propagation

Andreas Demuth
Roberto E. Lopez-Herrejon
Alexander Egyed

Johannes Kepler University Linz
Institute for Systems Engineering and Automation
{andreas.demuth|roberto.lopez|alexander.egyed}@jku.at

September 30th, 2013
Co-Evolution Definition

"The change of an object triggered by the change of a related object."

Co-Evolution Definition

- "The change of an object triggered by the change of a related object."

Co-Evolution Definition

"The change of an object triggered by the change of a related object."

Co-Evolution Definition

"The change of an object triggered by the change of a related object."

Traditional Co-Evolution

MM \rightarrowconforms\ M
Traditional Co-Evolution
Traditional Co-Evolution
Traditional Co-Evolution

\[ \text{MM} \quad \text{conforms} \quad \text{M} \]

\[ \Delta \quad \Delta \]

\[ \text{MM'} \]
Traditional Co-Evolution
Ambiguous Co-Evolution

- Definition
- Motivation
- Vision
- Future Work
Ambiguous Co-Evolution
Ambiguous Co-Evolution

- **MM** conforms to **M**
- **MM'** conforms to **M'**
- **Δ** to **Δ₂**

Fixed strategy to determine model update.
One solution chosen automatically.
May produce unintended model.
Ambiguous Co-Evolution

- MM
- MM'
- M
- M'
- Δ
- Δ
- Δ'
- Δ'
- Δ'
- Δ'
- Δ'

Fixed strategy to determine model update
One solution chosen automatically
May produce unintended model
Ambiguous Co-Evolution

- Fixed strategy to determine model update
Ambiguous Co-Evolution

- Fixed strategy to determine model update
- One solution chosen automatically
Ambiguous Co-Evolution

- Fixed strategy to determine model update
- One solution chosen automatically
- May produce unintended model
Ambiguous Co-Evolution Example

[Diagram showing Metamodel, Domain, Component, Communication, Model, and possible Receivers]
Ambiguous Co-Evolution Example
Ambiguous Co-Evolution Example
Ambiguous Co-Evolution Example

Metamodel

- Domain
  - Component
    - Communication

Model

- X
  - X1
  - X2
- Y
  - Y1
  - Y2
  - Y3
  - Y4

Communication

/possible Receivers

September 30th, 2013
Ambiguous Co-Evolution Example
Co-Evolution through Consistent Change Propagation
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
   - Missing or incorrect co-evolution
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
   - Missing or incorrect co-evolution
   - Through continuous consistency checking and incremental rule management
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
   - Missing or incorrect co-evolution
   - Through continuous consistency checking and incremental rule management

2. Ensure correct co-evolution
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
   - Missing or incorrect co-evolution
   - Through continuous consistency checking and incremental rule management

2. Ensure correct co-evolution
   - Provide user guidance to find suitable adaptations
Co-Evolution through Consistent Change Propagation

1. Detect co-evolution failure
   - Missing or incorrect co-evolution
   - Through continuous consistency checking and incremental rule management

2. Ensure correct co-evolution
   - Provide user guidance to find suitable adaptations
   - Execute desired adaptations and update model
Consistent Co-Evolution

MM \xrightarrow{\text{conforms}} M
Consistent Co-Evolution

脾胃 R

conforms
Consistent Co-Evolution

MM conforms to M

R

Environment
Consistent Co-Evolution

MM \rightarrow M \rightarrow R \rightarrow \text{Environment}
Consistent Co-Evolution
Consistent Co-Evolution
Consistent Co-Evolution
Consistent Co-Evolution
Consistent Co-Evolution
Consistent Co-Evolution

MM

Δ

MM'

Δ

M

R

Δ

M'

R'

Environment

conforms

conforms

conforms

conforms
Consistent Co-Evolution Example
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.

September 30th, 2013
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
**Consistent Co-Evolution Example**

**R1:** Possible receivers must belong to same domain.
**R2:** Communication only between distinct domains.
**R3:** Possible receivers must be distinct.
**R4**: Communications must provide alt2.
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.

September 30th, 2013
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
**R3: Possible receivers must be distinct.**
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
**R3: Possible receivers must be distinct.**
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
**Consistent Co-Evolution Example**

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.

---

**September 30th, 2013**
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.

September 30th, 2013
R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.
Consistent Co-Evolution Example

R1: Possible receivers must belong to same domain.
R2: Communication only between distinct domains.
R3: Possible receivers must be distinct.
R4*: Communications must provide alt2.

Definition | Motivation | Vision | Future Work
--- | --- | --- | ---
Consistent Co-Evolution Example | | |
Benefits
Benefits

- Support of arbitrary metamodel changes
Benefits

- Support of arbitrary metamodel changes
- No categorization needed
Benefits

- Support of arbitrary metamodel changes
  - No categorization needed
- Guidance for complex co-evolution scenarios
Benefits

- Support of arbitrary metamodel changes
  - No categorization needed
- Guidance for complex co-evolution scenarios
- Avoidance of unintended results
Benefits

- Support of arbitrary metamodel changes
  - No categorization needed
- Guidance for complex co-evolution scenarios
- Avoidance of unintended results
- Based on constraints and specific model
Benefits

- Support of arbitrary metamodel changes
  - No categorization needed
- Guidance for complex co-evolution scenarios
- Avoidance of unintended results
- Based on constraints and specific model
- Support for external constraints
Implementation Status
Implementation Status

- Individual parts are implemented
Implementation Status

- Individual parts are implemented
  - Continuous consistency checking of arbitrary models
Implementation Status

- Individual parts are implemented
  - Continuous consistency checking of arbitrary models
  - Incremental constraint management with automatic constraint updates
Implementation Status

- Individual parts are implemented
  - Continuous consistency checking of arbitrary models
  - Incremental constraint management with automatic constraint updates
    - MoDELS’13 Foundations Track, Thursday 10:30AM
Implementation Status

- Individual parts are implemented
  - Continuous consistency checking of arbitrary models
  - Incremental constraint management with automatic constraint updates
    - MoDELS’13 Foundations Track, Thursday 10:30AM
  - Efficient derivation of fixes for inconsistencies
Implementation Status

- Individual parts are implemented
  - Continuous consistency checking of arbitrary models
  - Incremental constraint management with automatic constraint updates
    - MoDELS’13 Foundations Track, Thursday 10:30AM
  - Efficient derivation of fixes for inconsistencies
  - (Semi-)Automatic execution of selected fixes
Future Work

- Integrate partial implementations
Future Work

- Integrate partial implementations
- Validation with industrial metamodels and models
Future Work

- Integrate partial implementations
- Validation with industrial metamodels and models
- Evaluate support for common refactorings
Co-evolution of Metamodels and Models through Consistent Change Propagation

Andreas Demuth
Institute for Systems Engineering and Automation
Johannes Kepler University Linz, Austria
email: andreas.demuth@jku.at
www: http://www.jku.at/sea