Enabling Constraint-Driven Modeling through Transformation

Andreas Demuth
Roberto E. Lopez-Herrejon
Alexander Egyed

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

- Problem Statement
- Approach Overview
- Applications
- Discussion
Problem Statement
Problem Statement

:LightSwitch

Switch

LightSwitch
Problem Statement

LightSwitch

1: activate

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

Switch

LightSwitch

1: activate

Decision cannot be automated guaranteeing that only intended results are produced.
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

1: activate

from

s:SequenceDiagram!Message
to

t:ClassDiagram:Method(
   name <- s.name,
   owner <- getClass(s.receiver.className)
)

Target

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

1: activate

Message

from
	s:SequenceDiagram

to

t:ClassDiagram:Method(

  name <- s.name,
  owner <- getClass(
    s.receiver.className)
)

Target

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

1: activate

from
s:SequenceDiagram!Message
to
t:ClassDiagram:Method(
   name <- s.name,
   owner <- getClass(
      s.receiver.className)
)

Target

Switch

LightSwitch

activate ( )
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch
Problem Statement

Source

:LightSwitch

1: activate

from s:SequenceDiagram!Message to

t:ClassDiagram:Method(
   name <- s.name,
   owner <- getSuperClass(s.receiver.className)
)

Target

Switch

LightSwitch

Decision cannot be automated guaranteeing that only intended results are produced.

May 29th, 2012
Problem Statement

```
Source

:LightSwitch

1: activate

from
s:SequenceDiagram!Message
to

Message

1: activate

Switch

:LightSwitch

from

S:SequenceDiagram!Message
to

T:ClassDiagram:Method(
   name <- s.name,
   owner <- getSuperClass(s.receiver.className)
)

Target

Switch

LightSwitch
```
Problem Statement

Source

:LightSwitch

from

s:SequenceDiagram!Message
to

t:ClassDiagram:Method(
   name <- s.name,
   owner <- getSuperClass(
      s.receiver.className)
)

Target

Switch

activate ( )

LightSwitch

May 29th, 2012
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch

Decision cannot be automated guaranteeing that only intended results are produced.
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

activate ( )

LightSwitch

activate ( )
Problem Statement

Source

:LightSwitch

1: activate

Target

Switch

activate ( )

LightSwitch

activate ( )
Problem Statement

Decision cannot be automated
Decision cannot be automated
guaranteeing that only intended results are produced
Constraint-Driven Modeling
Constraint-Driven Modeling

1: activate

Switch

LightSwitch
Constraint-Driven Modeling

LightSwitch must provide `activate()`

Switch

LightSwitch
Constraint-Driven Modeling

LightSwitch must provide `activate()`
Constraint-Driven Modeling

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch
Constraint-Driven Modeling

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch

from s:SequenceDiagram!Message

to
t:ConstraintModel!Constraint(
  context <- s.receiver.className,
  inv <- "self.providedMethods->
  exists(m|m.name='"+s.name+"')"
)

Constraint generation can be automated.
Guidance for resolving inconsistencies can be provided.

May 29th, 2012 4/20
Constraint-Driven Modeling

Source

:LightSwitch

1: activate

Target

Switch

self.providedMethods ->exists(m|m.name='activate')
Constraint-Driven Modeling

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch

self.providedMethods
->exists(m|m.name='activate')

Constraint generation can be automated
Guidance for resolving inconsistencies can be provided

May 29th, 2012 4/20
Constraint-Driven Modeling

```
self.providedMethods -> exists(m | m.name = 'activate')
```

- Constraint generation can be automated
Constraint-Driven Modeling

- **Source**
  - :LightSwitch
  - 1: activate

- **Target**
  - Switch
  - LightSwitch

```java
self.providedMethods ->exists(m|m.name='activate')
```

- Constraint generation can be automated
Constraint-Driven Modeling

Source

:LightSwitch

1: activate

Add method to LightSwitch
Add method to Switch
...

Target

Switch

LightSwitch

self.providedMethods
->exists(m|m.name='activate')

Constraint generation can be automated
Constraint-Driven Modeling

- Add method to LightSwitch
- Add method to Switch
- self.providedMethods ->exists(m|m.name='activate')

Constraint generation can be automated

- guidance for resolving inconsistencies can be provided
Constraint-Driven Modeling

- Constraint generation can be automated
- guidance for resolving inconsistencies can be provided

Source: LightSwitch

1: activate

Add method to LightSwitch
Add method to Switch
...

self.providedMethods ->exists(m|m.name='activate')

Target:

Switch
activate ()

LightSwitch
Constraint-Driven Modeling

Constraint generation can be automated
- guidance for resolving inconsistencies can be provided
Constraint-Driven Modeling

- Constraint generation can be automated
- Guidance for resolving inconsistencies can be provided
Providing Guidance

- Based on constraint and target metamodel
- Strategy for finding options is required
  - defined with constraint
  - based on constraint language
- Found options are endogenous transformations of target model
- Choose an option = execute transformation
Application: Incomplete Information

How are ambiguities handled?

\[ S \xrightarrow{t} T \]

\[ S \xrightarrow{t} C \rightsquigarrow T \]

(a) Transformation.

(b) Constraining.
Application: Incomplete Information

How are ambiguities handled?

(a) Transformation.

(b) Constraining.
Application: Incomplete Information

How are ambiguities handled?

\[ S \xrightarrow{t} T \]

(a) Transformation.

\[ S \xrightarrow{t} C \sim T \]

(b) Constraining.

May 29th, 2012
Application: Incomplete Information

- How are ambiguities handled?

\[ S \xrightarrow{t} T \]

\[ S \xrightarrow{t} C \sim T \]

(a) Transformation.  
(b) Constraining.
Application: Incomplete Information

- How are ambiguities handled?

\[ S \xrightarrow{t} T \quad S \xrightarrow{t} C \sim T \]

(a) Transformation.

(b) Constraining.
Application: Incomplete Information

How are ambiguities handled?

$S \xrightarrow{t} T$

$S \xrightarrow{t} C \rightsquigarrow T$

(a) Transformation.

(b) Constraining.
Application: Incomplete Information

How are ambiguities handled?

- Transformation:
  \[ S \xrightarrow{t} T \]

- Constraining:
  \[ S \xrightarrow{t} C \leadsto T \]

(a) Transformation.

(b) Constraining.
Application: Incomplete Information

How are ambiguities handled?

\[ S \xrightarrow{t} T \]

(a) Transformation.

\[ S \xrightarrow{t} C \leadsto T \]

(b) Constraining.
Application: Target Model Updates

- How are model updates handled?
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

Target

Switch

LightSwitch

activate ( )

Standard transformation
Application: Target Model Updates

How are model updates handled?

Standard transformation

Source

Target

Switch

activate ()

LightSwitch

activate ()
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

Target

Switch'

activate ( )

LightSwitch'

Standard transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch'

activate ( )

LightSwitch'

Standard transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch'

activate ( )

LightSwitch'

activate ( )
deactivate ( )

Standard transformation
How are model updates handled?

Source:

- :LightSwitch
- 1: activate
- 2: deactivate

Target:

- Switch'
- activate ( )
- LightSwitch'
- activate ( )
- deactivate ( )

Standard transformation

Application: Target Model Updates
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch'

activate ( )

LightSwitch'

activate ( )
deactivate ( )

Standard transformation
Application: Target Model Updates

How are model updates handled?

Source
:LightSwitch

Target
Switch

LightSwitch

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

Target

Switch

activate ( )

LightSwitch

self.providedMethods->exists(mm.name='activate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source:

:LightSwitch

1: activate

2: deactivate

Target:

Switch

activate ( )

LightSwitch

self.providedMethods->exists(mm.name='activate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch

activate ( )

LightSwitch

self.providedMethods->exists(mm.name='activate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch

activate ( )

LightSwitch

Constraint transformation

self.providedMethods->exists(mm.name='activate')

self.providedMethods->exists(mm.name='deactivate')
Application: Target Model Updates

How are model updates handled?

Source

1: activate

2: deactivate

Target

Switch

activate ()

LightSwitch

self.providedMethods->
exists(mm.name='activate')

self.providedMethods->
exists(mm.name='deactivate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch

activate ( )

deactivate ( )

LightSwitch

self.providedMethods->exists(mm.name='activate')

self.providedMethods->exists(mm.name='deactivate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

Source

:LightSwitch

1: activate

2: deactivate

Target

Switch

activate ( )
deactivate ( )

LightSwitch

self.providedMethods->exists(mm.name='activate')

self.providedMethods->exists(mm.name='deactivate')

Constraint transformation
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

- How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

- How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Target Model Updates

- How are model updates handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.
(b) Constraining.
Application: Rule Scheduling

- How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

- How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

- How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

- How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

- How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Rule Scheduling

How are different orders of rule execution handled?

(a) Transformation.

(b) Constraining.
Application: Different Source Metamodels

How can different source models be used for constructing a single target?

(a) Constraining.
Application: Different Source Metamodels

How can different source models be used for constructing a single target?

(a) Constraining.
Application: Different Source Metamodels

How can different source models be used for constructing a single target?

(a) Constraining.
Application: Different Source Metamodels

- How can different source models be used for constructing a single target?

(a) Constraining.
Application: Different Source Metamodels

How can different source models be used for constructing a single target?

(a) Constraining.
Application: Different Source Metamodels

How can different source models be used for constructing a single target?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Application: Bidirectionality

- How are two models synchronized?

(a) Constraining.
Applications

- Incomplete Information
- Target model updates
- Rule scheduling
- Different source metamodels
- Model synchronization
Approach Characteristics I

- Help designers getting the desired model vs. automating non-automatable tasks
- Constraints generated automatically through transformations
- User guidance based on constraints and target metamodel
- Transformation options for target model derived from constraints
- No automatic changes of target model
Approach Characteristics II

- Delay of decisions
  - from metamodel/transformation design time
  - to modeling time
- Resulting models may be different from conventional transformation
  - based on additional domain knowledge
Constraint Characteristics

- Constraints are additive and independent
  - can be added at any time
  - without scheduling issues
  - can come from different source models
- Defining desired characteristics requires less information than defining a specific solution
  - constraints are easy to generate
- Provide information even without generated guidance
Validation

- Working prototype
  - Custom incremental transformation engine
  - UML metamodel
  - OCL-like constraints
  - Models: 20 industrial (up to 160k elements), 10 synthetic
  - Model/Analyzer incremental consistency checker
Performance

- Test 1 – Ambiguous transformations
- Test 2 – Merges for multiple sources
- Add/remove elements (e.g., a message) to/from source
- Transformation and constraint validation measured
Applicability

- Approach is generic and not limited to
  - a specific transformation engine
  - a specific constraint language
  - specific source and target metamodels
  - a specific consistency checking technique
Future Work

- Usability studies and case studies
- User guidance
- Automated fixes
- Constraints contradictions / overconstraining
Constraint-Driven Modeling through Transformation

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