A Constraint Modeling Framework for Domain-Specific Languages

Patrick Pschorn, Pablo Antonino Oliveira, Andreas Morgenstern, Thomas Kuhn

University of Kaiserslautern
Fraunhofer IESE

Splash DSM’19, Athens, Greece
Classic Model Driven Development
Classic Model Driven Development

Need to satisfy:
- Explicit design
- Domain-specific rules
Today: Architecture View Frameworks

Domain Model

Need to satisfy:
- Explicit design
- Domain-specific rules

Fraunhofer Embedded Modeling View Framework [1]
Domain-Specific Rules

Example: Domain Expert derives the following criterion

Safety Requirement Completeness and Consistency Criterion SRComp1 [1]

*Every Safety Requirement (SR) describes mitigation strategies for failures that are described in at least one failure propagation model (FPM).*

Domain Expert
Domain-Specific Rules

Example: Implementation using OCL [2] could look like this:

```
— classes
class SafetyRequirement
  attributes
    name: String
    SIL: String
end

class FailurePropagationModel
  attributes
    name: String
end

— associations
association isMotivatedBy between
  SafetyRequirement [1..*] role sr
  FailurePropagationModel [1..*] role fpm
end

— constraint
context SafetyRequirement inv SRatleastOneFPM:
  SafetyRequirement.allInstances().size() > 0
```

Advantages
• Technical
• Unambiguous

Disadvantages
• Complicated
• Effort Consuming
• Limited to UML, cannot query tool API

SRComp1 OCL Notation [1]
Related Work on Constraint Specification

**General Purpose**
- OCL [2]
- EVL [3]
- (Set Theory [4])

*Positive:* Expressiveness, Automation

*Negative:* Understandability, Customizability

**Template-based**
- MetaEdit+ Forms [5]
- Excel Templates [6]

*Positive:* Simple & Fast

*Negative:* Fixed Pattern

**Diagram-based**
- Constraint Diagrams [7]
- PREEvision rule editor [8]

*Positive:* Usability, Flexible

*Negative:* Ambiguity, Limited for complex rules
Related Work on Constraint Specification

**General Purpose**
- OCL [2]
- EVL [3]
- (Set Theory [4])

*Positive:* Expressiveness, Automation

*Negative:* Understandability, Customizability

**Template-based**
- MetaEdit+ Forms [5]
- Excel Templates [6]

*Positive:* Simple & Fast

**Diagram-based**
- Constraint Diagrams [7]
- PREEvision rule editor [8]

*Positive:* Usability, Flexible

*Negative:* Ambiguity, Limited for complex rules
Related Work on Constraint Specification

General Purpose
- OCL [2]
- EVL [3]
- (Set Theory [4])

Positive:
Expressiveness, Automation

Negative:
Understandability, Customizability

Template-based
- MetaEdit+ Forms [5]
- Excel Templates [6]

Positive:
Simple & Fast

Negative:
Fixed Pattern

Diagram-based
- Constraint Diagrams [7]
- PREEvision rule editor [8]

Positive:
Usability, Flexible

Negative:
Ambiguity, Limited for complex rules
Related Work

General Purpose
- OCL [2]
- EVL [3]
- (Set Theory [4])

Template-based
- MetaEdit+ Forms [5]
- Excel Templates [6]

Diagram-based
- Constraint Diagrams [7]
- PREEvision rule editor [8]

Qualitative Requirements on constraint specification facilities
- Understandable and quick specification
- Expression of conditional and logically nested constraints
Functional Requirements

1. The user needs to be able to specify restrictions on the (inter-element) **relationships** between model elements.

2. The user needs to be able to specify restrictions on the **set of legal values** that the model element properties can have.

3. The user needs to be able to specify restrictions on the number of **(visual) occurrences** of model elements in certain diagram view types.

*(Collected from literature and during the development of constraints for the Embedded Modeling Profile [6] used in our projects.)*
The Constraint Modeling Meta-Metamodel

---

**classes**

```ocl
class SafetyRequirement
attributes
  name: String
  SIL: String
end

class FailurePropagationModel
attributes
  name: String
end
```

---

**associations**

```ocl
association isMotivatedBy between SafetyRequirement [1] role sr
  FailurePropagationModel [1..*] role fpm
end
```

---

**constraint**

```ocl
context SafetyRequirement inv SRatleastOneFPM:
  SafetyRequirement.allInstances().forall(fpm -> fpm.size() > 0)
```

---

**Generate**

Constraint Relation

SRComp1 OCL Notation [1]
The Constraint Modeling Meta-Metamodel

Filter

Constraint Relation

SRComp1 OCL Notation [1]

```plaintext
— classes
class SafetyRequirement
attributes
   name: String
   SIL: String
end

class FailurePropagationModel
attributes
   name: String
end

— associations
association isMotivatedBy between
   SafetyRequirement [1] role sr
   FailurePropagationModel [1..*] role fpm
end

— constraint
class SafetyRequirement inv SRatleastOneFPM: SafetyRequirement . allInstances () -> 
   forall (fpm -> size () > 0)
```
The Constraint Modeling Meta-Metamodel

Filter

Logical Gate

Constraint Relation

SRComp1 OCL Notation [1]
The Constraint Modeling Meta-Metamodel
Constraint Modeling and the DSM Process

Constraint Modeling Language

Domain Model

Validate

Domain-Specific Modeling Profile

Stereotype Elements

Executable Artifacts

Generator

Filter Execution Behavior

Constraint Modeling Diagram

Stereotype Elements

Constraint Modeling and the DSM Process

Domain Model

Validate

Executable Artifacts

Generator

Filter Execution Behavior

Constraint Modeling Diagram

Stereotype Elements

Domain-Specific Modeling Profile
Constraint Modeling and the DSM Process

Domain-Specific Modeling Profile

Stereotype Elements

Constraint Modeling Diagram

Filter Execution Behavior

Generator

Executable Artifacts

Validate

Domain Model

Stereotype Elements

Constraint Relations
- has exclusive relationship to
- has n-ary relationship to
- has limited relationships
- property value compares to

Filters
- property value satisfies
- only in this diagram
Case Study: Completeness of Safety Requirements

SRCompC1 – C3 by Antonino [1], Original Notion
Case Study: Completeness of Safety Requirements

SRCompC1 – C3, Constraint Modeling
Language Representation
Contributions

• Identified requirements for constraint specification facilities
• Proposed Constraint Modeling Framework
• Provided concept design for a tool and a compiler that translates CMD into executable UML Constraints

Findings

• Language independence makes rule semantics more powerful than simple OCL.
• Abstract, graphical representation yields increased understandability and usability for constraints on completeness and consistency of architecture models.
References

References


Thank you