Generic Upsilon Transformations

A. Hovsepyan, D. Delanote, S. Van Baelen, W. Joosen, Y. Berbers
Katholieke Universiteit Leuven, Belgium

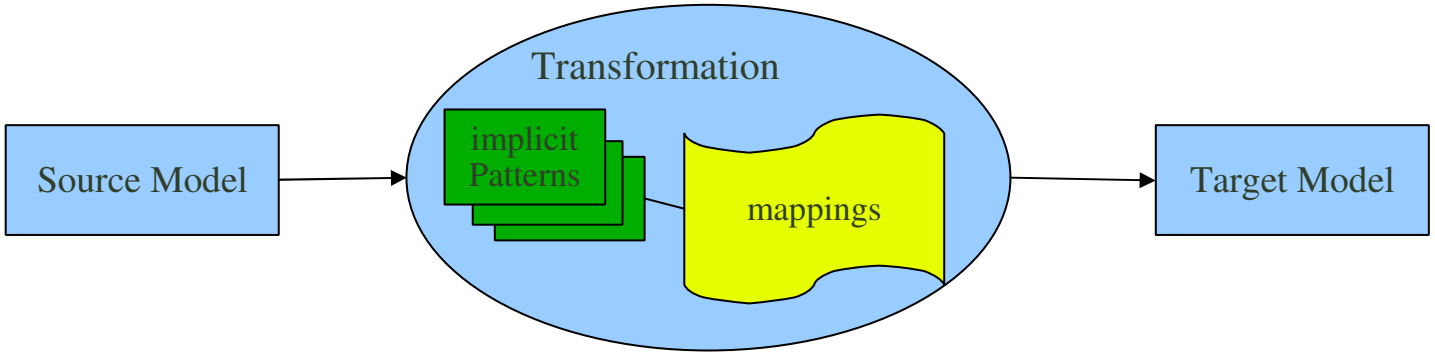
KULeuven
DistriNet
Research Group

Outline

- Introduction
- Generic Upsilon Transformations
  - description
  - basic mappings
  - extra constructs
- GUT Tool
- Discussion and future work

Introduction

- Current transformation approaches



- Disadvantages:
  - patterns hard-coded in transformations
    - difficult to maintain
      - if pattern changes even slightly transformation should be modified
    - non-reusable from one tool to another
  - one transformation is realized as a number of patterns applied simultaneously on source model
    - transformation reusability goes down

Introduction

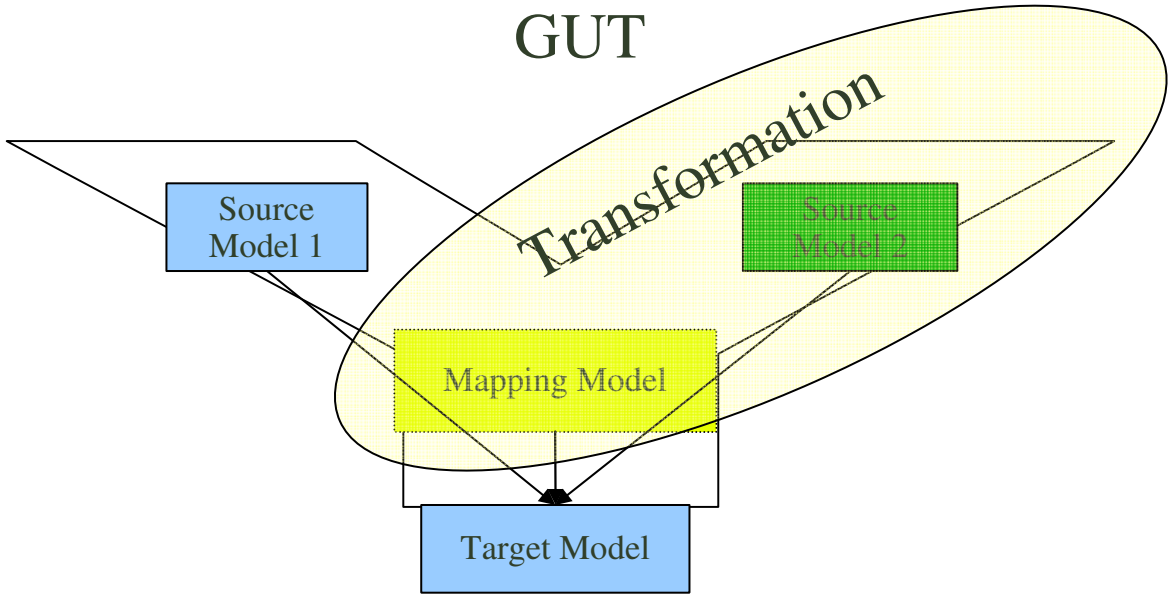
- Transformation reusability
  - Why
    - increase productivity in an MDD process
    - reliability
    - economy of scale
  - What reusable means
    - generic
    - domain-independent
    - sufficiently fine-grained
    - library of reusable transformations
- Independence from transformation languages (ATL, MTF, etc.)
  - graphical specification
  - high level transformation primitives

Outline

- Introduction
- Generic Upsilon Transformations
  - description
  - basic mappings
  - extra constructs
- GUT Tool
- Discussion and future work

Generic Upsilon Transformations

- Generic Upsilon Transformations (GUT)
  - model transformation as pattern (Source Model 2)
  - graphical specification of how source model is mapped onto transformation pattern (Mapping Model)
  - use generic transformation engine



Generic Upsilon Transformations

- Generic Upsilon Transformations (GUT)
  - Each transformation represented as a Pattern Model (UML)
    - fine-grained transformations
    - raise abstraction level
  - Mapping Model (UML)
    - generic & domain-independent transformations
      - same transformation pattern can be applied in different contexts using a different Mapping Model
    - raise abstraction level
  - GUT Transformation = Pattern Model + Mapping Model

Generic Upsilon Transformations

- Two types of transformations
  - Model composition (horizontal transformation)
    - Source Model 1 and Source Model 2 are UML models
    - Mapping Model specifies how they should be composed/merged
  - Pattern mapping (vertical transformation)
    - Source Model 1 is a model
    - Source Model 2 is a pattern model (generic, non-parameterized)
    - Mapping Model describes how the pattern is applied on Source Model 1
- All models are UML2 class diagrams
  - GUT uses a very small subset of UML2 metamodel and can be implemented in other modeling languages (e.g. Ecore, MOF)

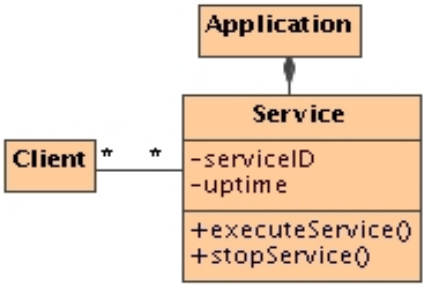
Generic Upsilon Transformations

- Specify 3 input models
  - Source Model 1 (UML)
  - Source Model 2 (UML)
  - Mapping Model (UML class diagram + GUT UML profile)
- Several mapping strategies available for mapping source models
  - filter
  - merge
  - copy
  - move
  - duplicate
  - rename
  - remove
- Elements not in Mapping Model are copied to Target Model

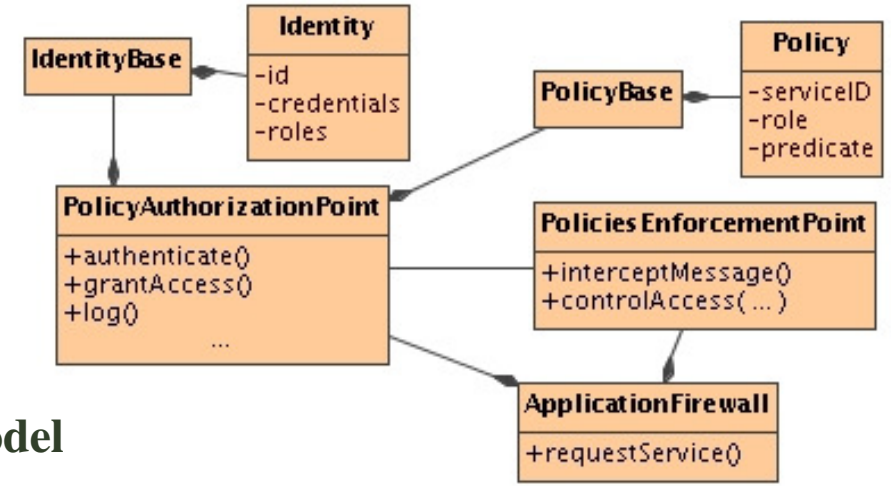
Generic Upsilon Transformations

- Mapping strategies: **filter**
  - Remap an association through a class

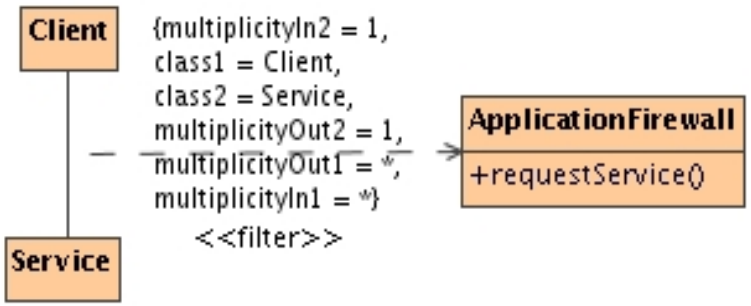
Source Model 1



Source Model 2



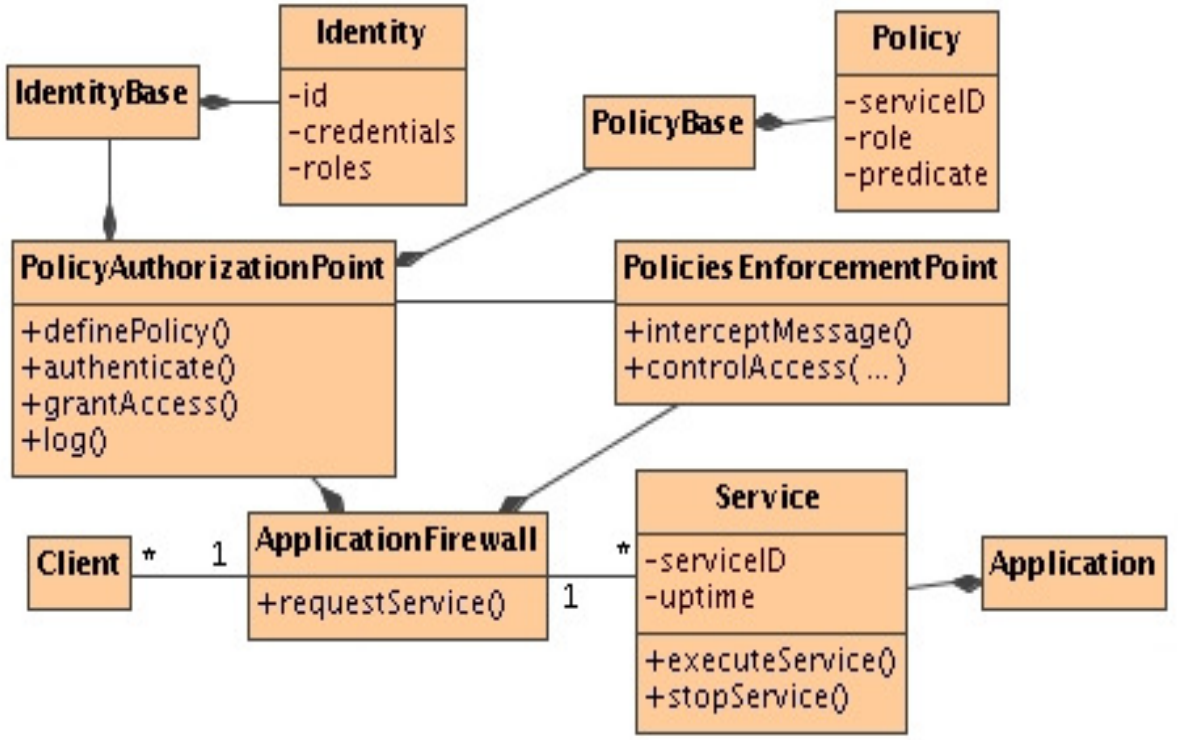
Mapping Model



Generic Upsilon Transformations

- Mapping strategies: **filter**

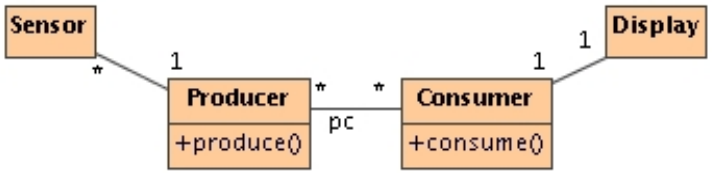
Target Model



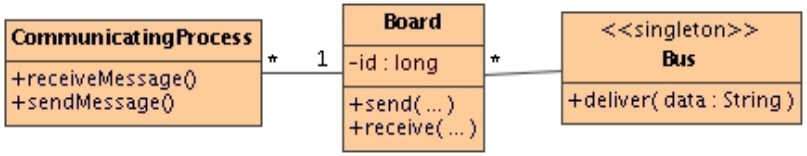
Generic Upsilon Transformations

- Mapping strategies : **merge**
  - Merge two given elements from source models
  - New element in Target Model combines properties of source entities

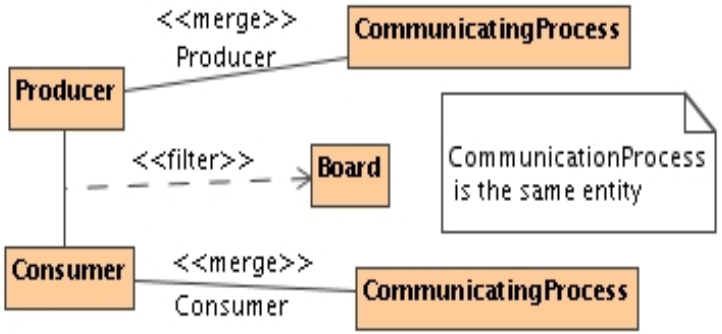
Source Model 1



Source Model 2



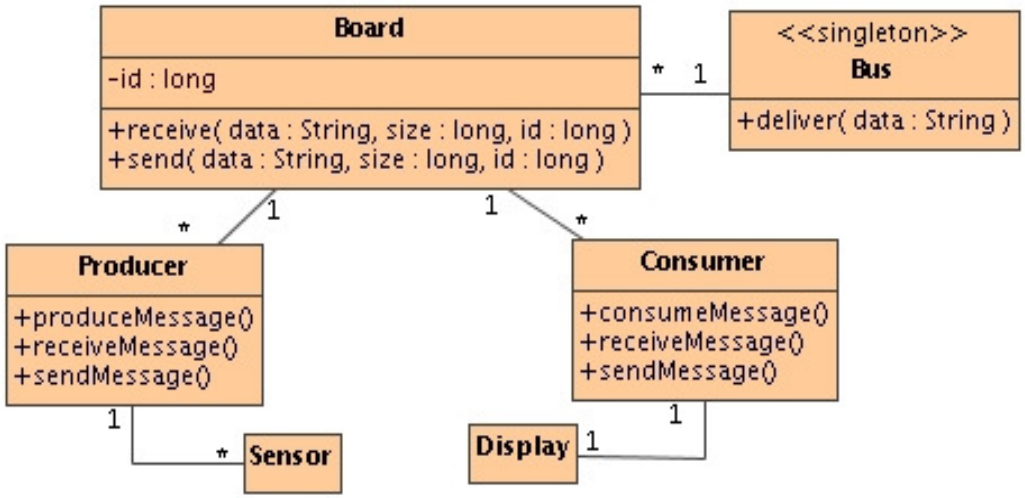
Mapping Model



Generic Upsilon Transformations

- Mapping strategies : **merge**

Target Model

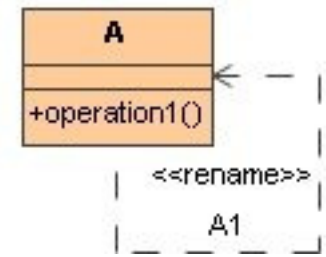


Generic Upsilon Transformations

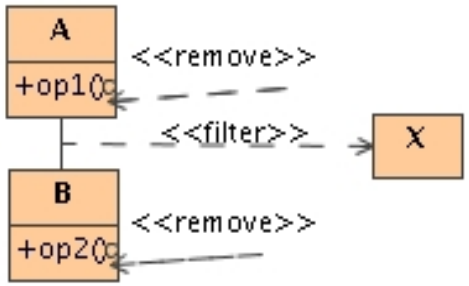
- Mapping strategies: property mapping
  - Patterns might refactor properties of classes by moving/copying them to another class
    - **copy**
    - **move**
- Mapping strategies : **duplicate**
  - duplicate classes or properties of a given class

Generic Upsilon Transformations

- Mapping strategies : **rename**
  - rename entities during mapping
    - use <<rename>> dependency to self on the mapping model



- Mapping strategies : **remove**
  - composition might make a model element superfluous
    - use <<remove>> dependency to self on the mapping model



Outline

- Introduction
- Generic Upsilon Transformations
  - description
  - basic mappings
  - extra constructs
- GUT Tool
- Discussion and future work

GUT Tool

- Tool implementation
  - Eclipse technology
  - Eclipse Modeling Framework (EMF)
  - Ecore implementation of UML
  - GUT UML Profile
  - Transformation engine: ATL

Outline

- Introduction
- Generic Upsilon Transformations
  - description
  - basic mappings
  - extra constructs
- GUT Tool
- Discussion and future work

Discussion and future work

- Library of patterns
  - with pre-defined mappings
  - how to store transformation/patterns in the library
  - how to search fo a given transformation/pattern
- Use GUT approach on broader embedded system case-study
  - to what degree can GUT support embedded system development