



Model Transformation By Example

Dániel Varró

Budapest University of Technology and Economics

Motivation

- **Model transformation: a key problem for MDD**
 - Model analysis, refactoring, model synchronization
code generation, deployment, etc.
 - Wide range of model transformation languages and tools
- **Who are transformation designers?**
 - Transformation experts
 - Need to understand the transformation language
 - Need to understand the source and target domains
 - There are not that many of them
 - **Domain engineers**
 - understand the source and target domains / languages
 - no skills in the model transformation language

Idea

- **Analogy from the XML world**

- Connect sample source and target XML documents
- Generate XSLT scripts automatically



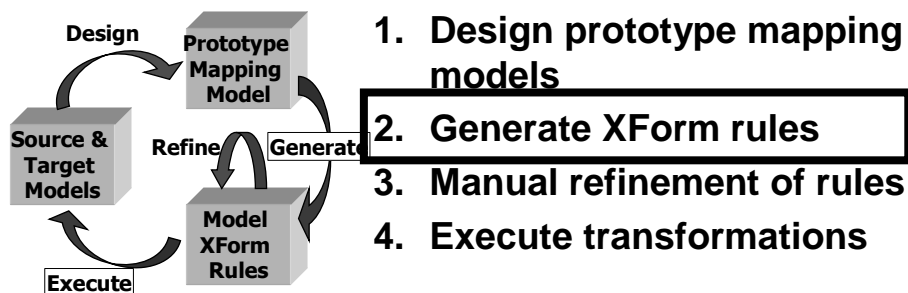
- **Proposal: Model Transformation by Example**

- Create interrelated prototypical model pairs (source – reference/trace – target)
- Generate model transformation rules automatically

- **Advantage**

- Transformation designers use their own domains

Overview of the Approach



Budapest University of Technology and Economics 5

(MT by Example) by Example The Object-Relational Mapping

Fault Tolerant Systems Research Group MODELS 2006, Genova, Italy

Budapest University of Technology and Economics 6

Object-Relational Mapping

The diagram illustrates the mapping between an object-oriented model and a relational database schema.

Object Model (Left):

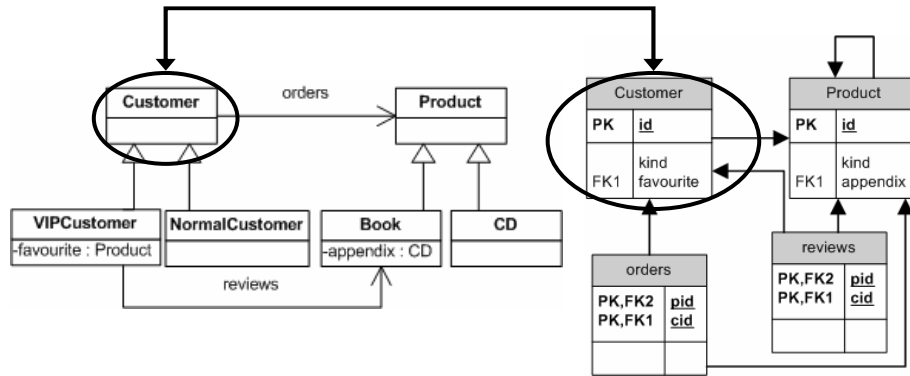
- Customer** (Superclass)
 - VIPCustomer** (Subclass): -favourite : Product
 - NormalCustomer** (Subclass)
- Product** (Superclass)
 - Book** (Subclass): -appendix : CD
 - CD** (Subclass)
- Relationships:**
 - orders:** Association between Customer and Product.
 - reviews:** Association between Customer and Book.

Relational Schema (Right):

- Customer Table:**
 - PK: id
 - FK1: kind favourite
- Product Table:**
 - PK: id
 - FK1: kind appendix
- orders Table:**
 - PK,FK2: pid
 - PK,FK1: cid
- reviews Table:**
 - PK,FK2: pid
 - PK,FK1: cid

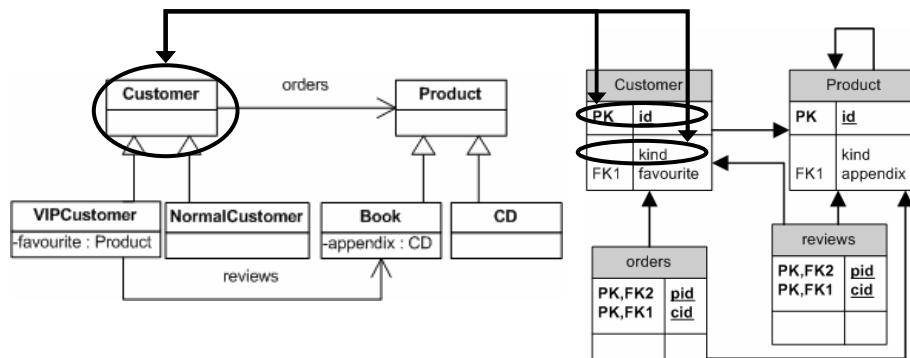
Fault Tolerant Systems Research Group MODELS 2006, Genova, Italy

Informal rules of the transformation



Each top-level class is projected into a DB table

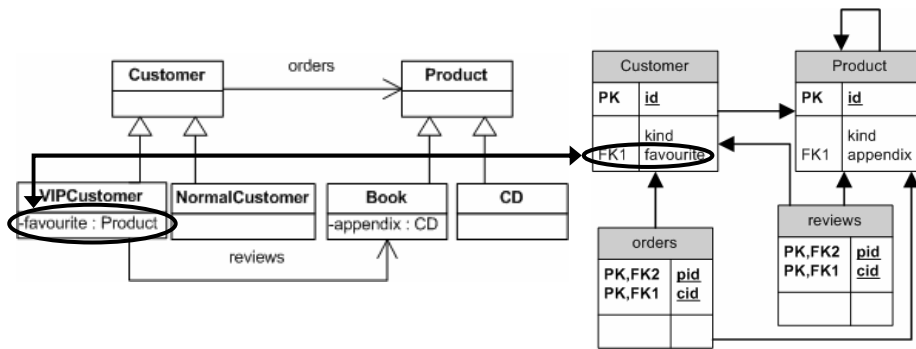
Informal rules of the transformation



Two additional columns are derived for each top-level class

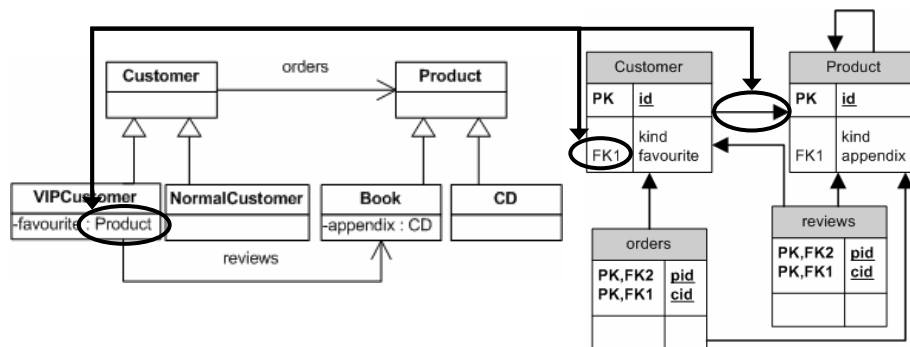
- a unique identifier (primary key),
- the type information of instances

Informal rules of the transformation



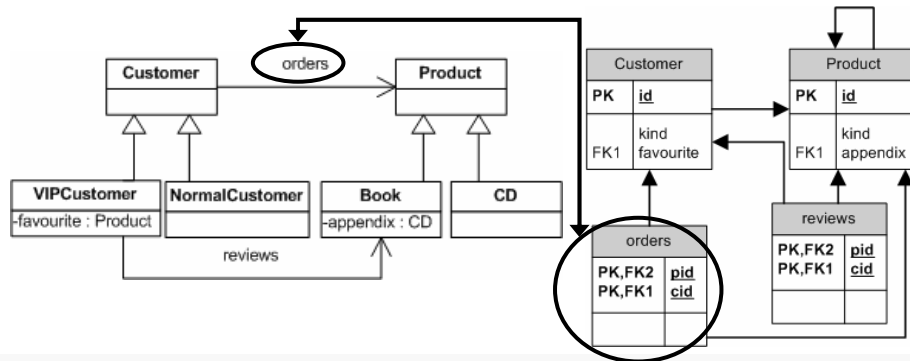
Each attribute of a class will appear as columns in the table related to the top-level ancestor of the class
 (The type of an attribute is restricted to user defined classes)

Informal rules of the transformation



The structural consistency of valid instances in columns is maintained by foreign key constraints

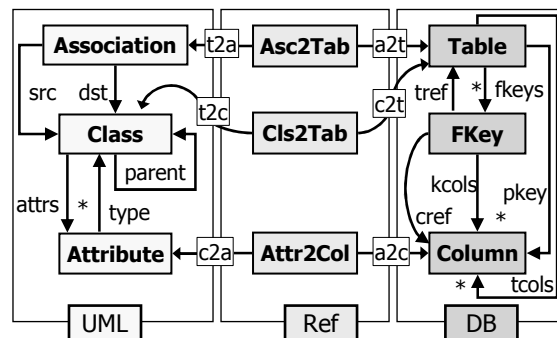
Informal rules of the transformation



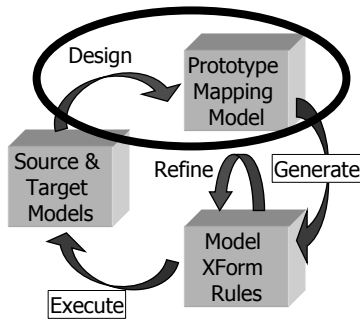
Each association is projected into a table with two columns

- pointing to the tables related to the source and the target classes
- consistency insured by foreign key constraints

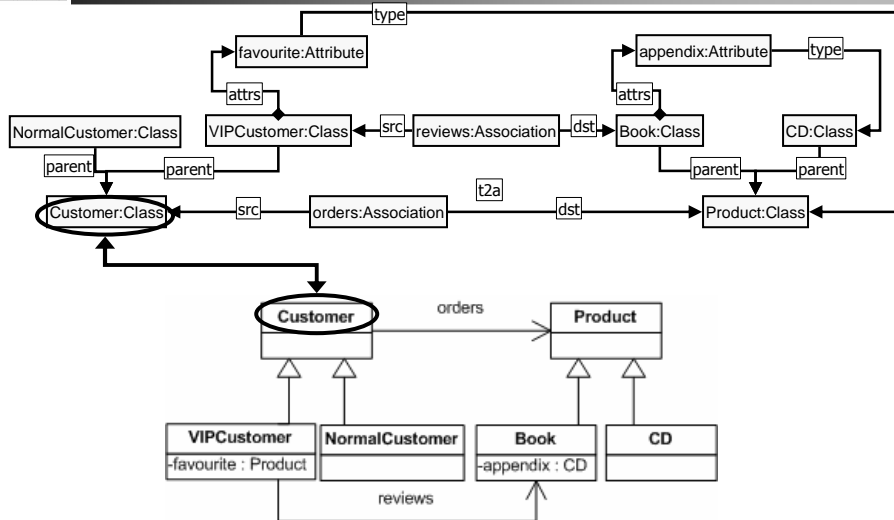
Metamodels of the problem

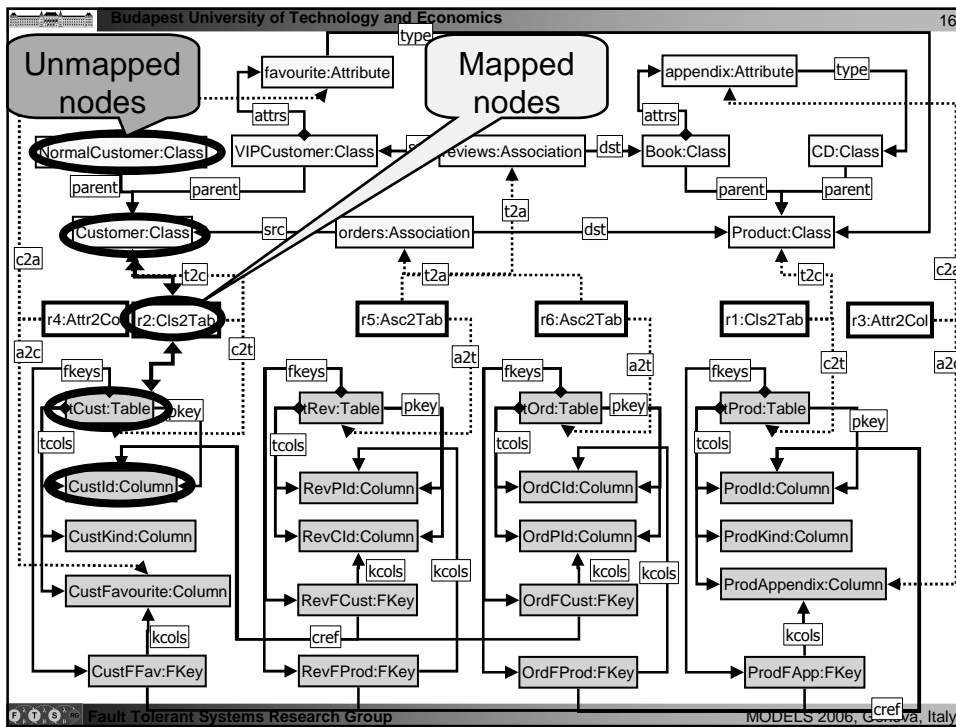
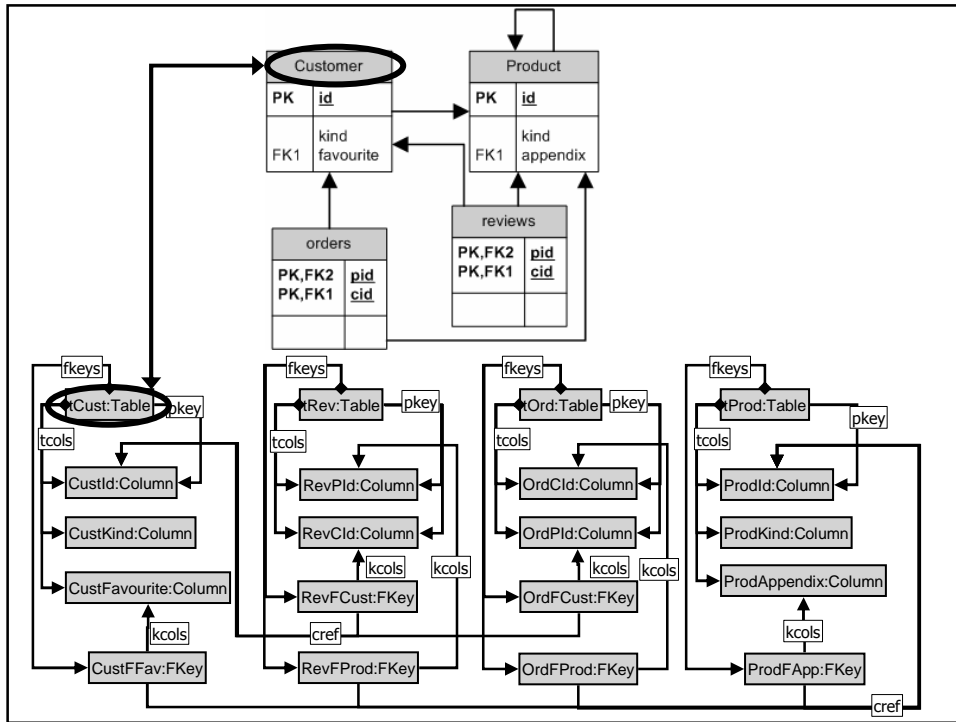


Step 1: Create Prototype Mapping Models



- **Prototype mapping model (PMM)**
 - Any pair of *src* and *trg* models
 - Interrelated by references
 - Capture critical situations of the transformation problem
- **Properties of PMMs**
 - small models
 - may serve as test cases later





Budapest University of Technology and Economics 17

Assumptions on Prototype Models

- Reference is also a graph
- Unique references.
- Existence of unmapped model elements.
- No merging transformations.
- Aggregation semantics (EMF).
- Correctness of prototype models (intentions).

Fault Tolerant Systems Research Group MODELS 2006, Genova, Italy

Budapest University of Technology and Economics 18

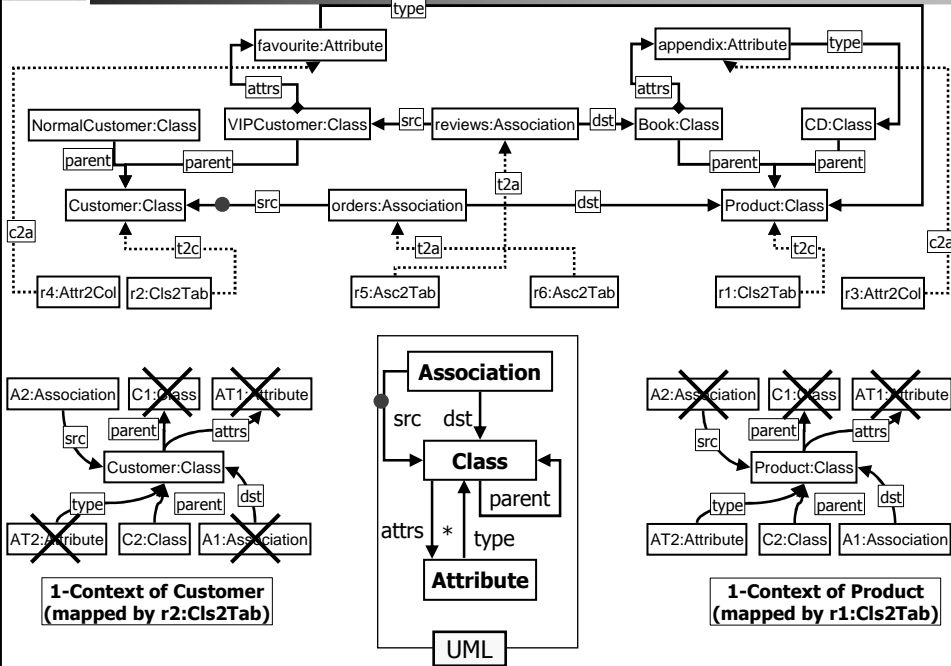
Step 2: Derive Transformation Rules

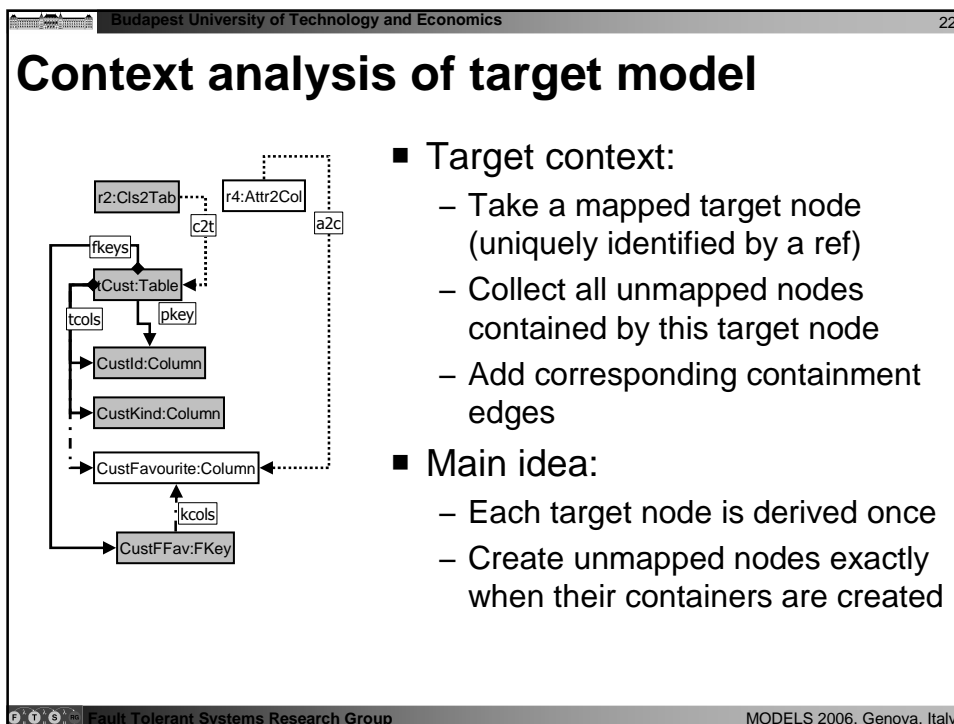
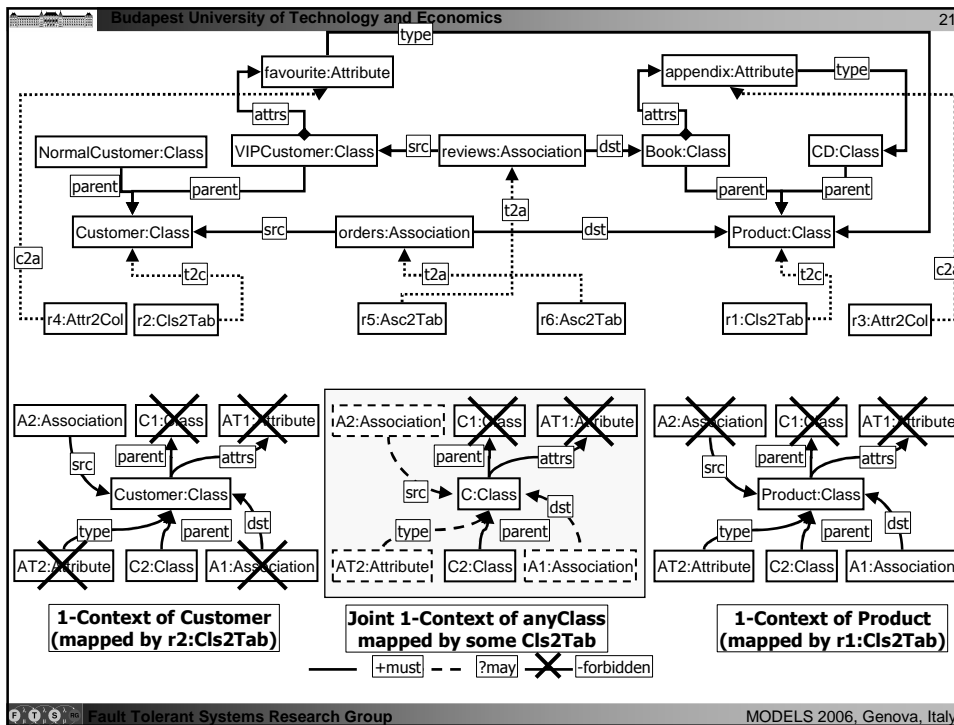
- **Phase 1: Creation of Mapped Target Nodes**
 - Context analysis of source model
 - Context analysis of target model
 - Derive transformation rules
- **Phase 2: Interconnecting target nodes**
 - Connectivity analysis
 - Derive transformation rules

Fault Tolerant Systems Research Group MODELS 2006, Genova, Italy

Context analysis of source model

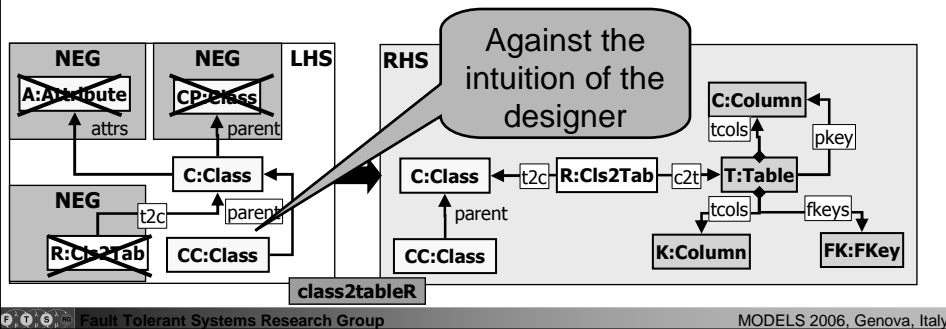
- **1-context (can be extended to n-contexts)**
 - Existence or non-existence of
 - Incoming or outgoing edges (all possible)
 - Related to a certain mapped node
- **Joint 1-context**
 - Generalization of 1-context
 - For all mapped source nodes of a certain type
 - Must / Forbidden / May edge
- **Check 1-context for unmapped elements**
 - Should be different from joint 1-context



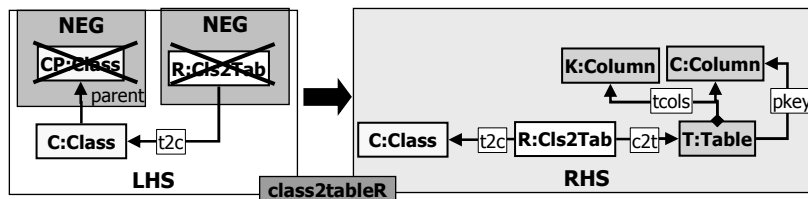
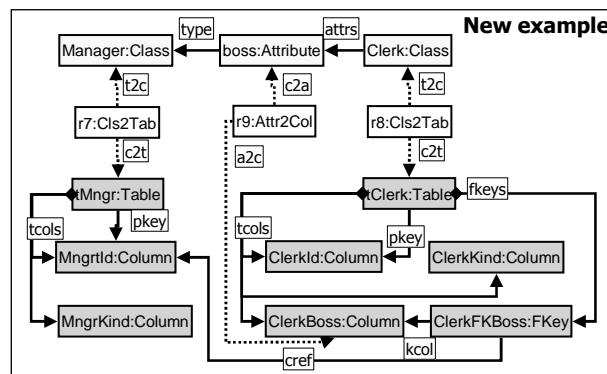


Derivation of model transformation rules

- **Graph transformation rules**
 - LHS: Precondition (with negative conditions)
 - RHS: Postcondition
- **Derivation of GT rules**
 - **Precondition:** *must* and *forbidden* in the joint *src* ctx
 - **Postcondition:** composition of joint *src* and *trg* context



Iterative development of rules



Connectivity analysis

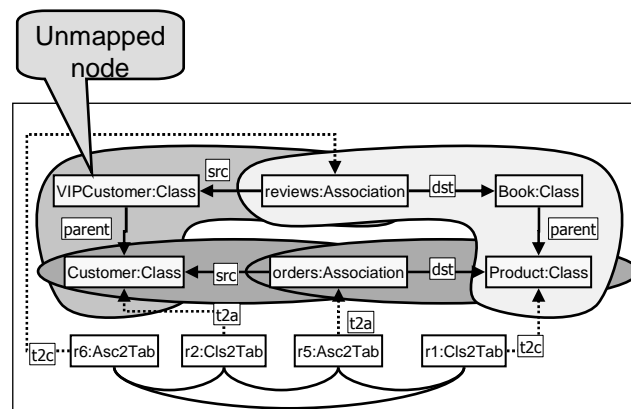
■ Goal:

- Derive the links between the corresponding target elements (created in the previous phase)

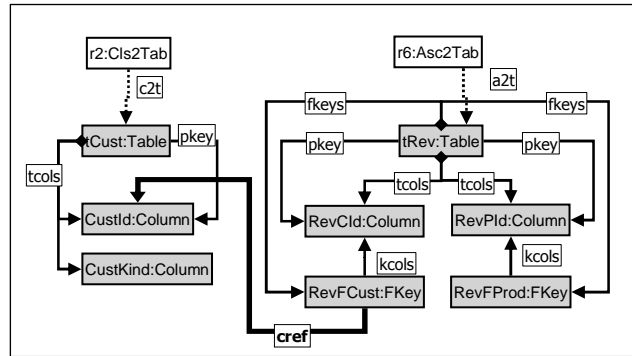
■ Connectivity analysis

- Investigate pairs (triples, etc.) of reference nodes
- Identify paths
 - between mapped source (target) nodes
 - leading through unconnected source (target) nodes

Connectivity analysis (Source model)

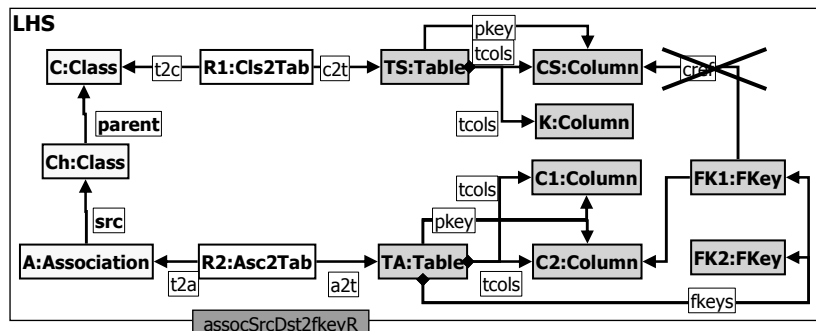


Connectivity analysis (Target model)



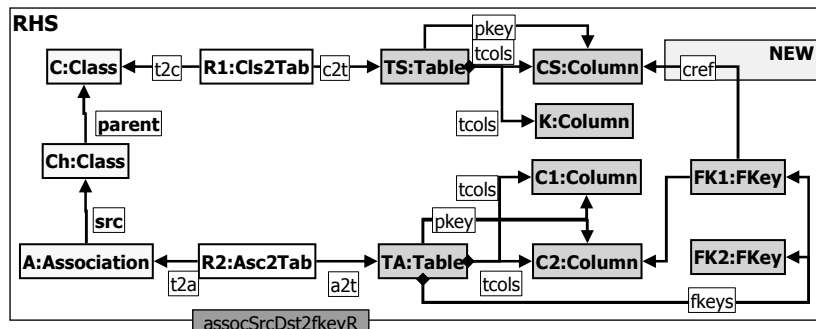
Derivation of transformation rules (for edges)

- **LHS:**
 - Source path
 - Reference node
 - Target context
- **Negative condition:**
 - Target path

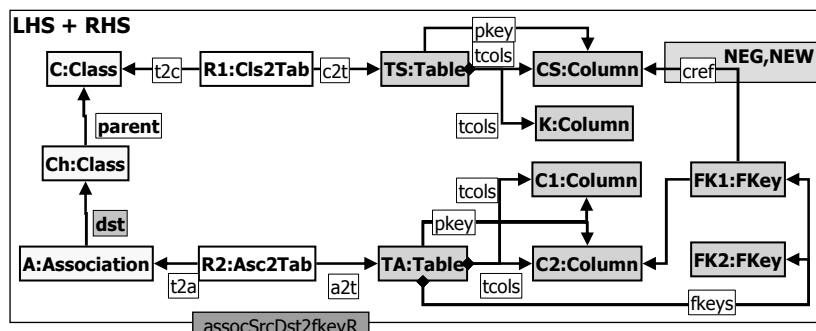


Derivation of transformation rules (for edges)

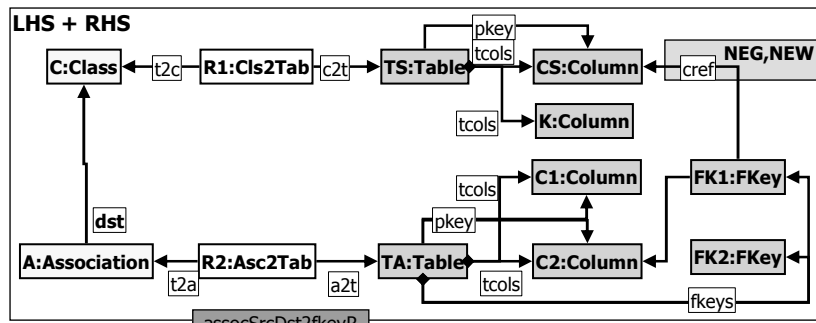
- **RHS:**
 - Full LHS
 - Target path (to be created)
- **Note:**
 - Generate GT rule only if target path always exists
 - Generate a GT rule for all source paths yielding the same target path



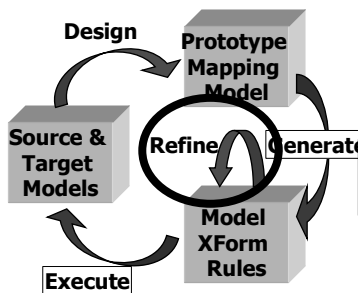
Derivation of transformation rules (for edges)



Derivation of transformation rules (for edges)



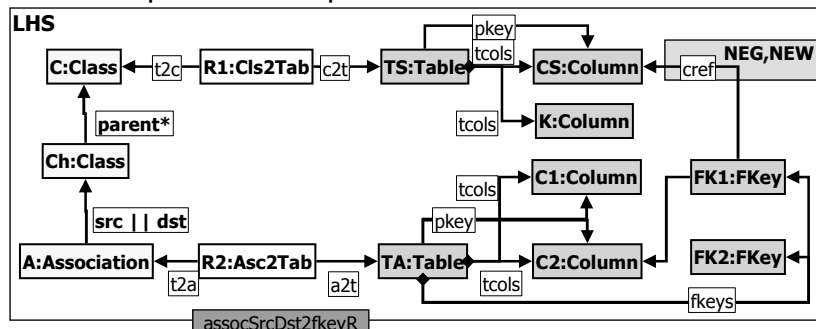
Overview of the Approach



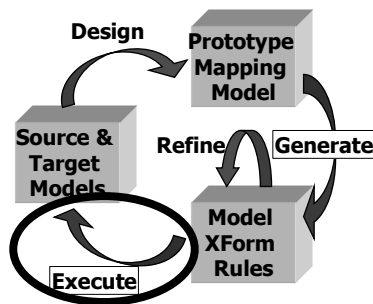
1. Design prototype mapping models
2. Generate XForm rules
3. Manual refinement of rules
4. Execute transformations

Step 3: Generalization of rules (manual)

- **Problem:**
 - Large number of transformation rules may be derived
- **Solution:**
 - Generalization of rules (manually? automatically?)
 - ➔ Path expressions in preconditions



Step 4: Execution of rules



- **Generated rules can be executed at any time**
 - Identify new test cases
 - Check if existing rules already properly handle them
- **In case of failure**
 - Extend PMMs with the new test case
 - Regenerate rules

Conclusions and Future work

- **Model Transformation by Example (MTBE)**
 - Create prototype mapping models to capture critical situations of the XForm problem
 - Derive transformation rules interactively
- **Initial experience**
 - Very promising new area
 - Initial stage of research
- **Ongoing activity**
 - Semantic foundations for MTBE
 - Tool support in VIATRA2