Software Transformations: A formalism to trace program modifications

Mathieu Lemoine  
Université de Montréal, DIRO,  
lemoinem@iro.umontreal.ca

Yann-Gaël Guéhéneuc  
École Polytechnique de Montréal, Software Engineering,  
yann-gael.gueheneuc@polymtl.ca

2009/01/16  
(4th MoSART Meeting)
1 Introduction
   ■ Context
   ■ Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   ■ Future Work
   ■ Conclusion

6 Bibliography
1 Introduction
   - Context
   - Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   - Future Work
   - Conclusion

6 Bibliography
Typical Problem: Documenting Software Evolution

Many people working at many abstraction levels on the same project at the same time.

Problem: How to keep track of each modification, in a way readable by every one (Developer, Analyst, Manager)?

Goal: reduce cost, ease communication, trace software evolution.
Current Solutions

- **Versioning Control Repositories (CVS, SVN, DARCS, ...)**: textual documents, only for developer
- **Model Driven/Reverse Engineering Architecture Tools (OMONDO, Ptidej [Gué05], ...)**: no simultaneous modifications of code and model.
- **Post-mortem analysis (detection of refactorings, entity matching [ACPT01])**: no live feedback, no rollback capabilities.
Development of a web server - Model

**First-draft, original specification**

**Actually created code modelization**
Development of a web server - Time line

A is Analyst, D is Developer, Each little number is a single modification.

1. Modification of Listener constructor (code modifications)
2. Creation of Item class (model modifications)
   Modifications sent to D
3. Renaming of Item to Page and implementation of Page (code modifications)
4. Creation of IAnswerer interface and its Factory (model modifications)
   Modifications sent to D
5. Implementation of the IAnswerer hierarchy (code modifications)
1 Introduction
   ■ Context
   ■ Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   ■ Future Work
   ■ Conclusion

6 Bibliography
Hoare Triples [Hoa69]

Representation of action having
- guards (pre-condition)
- effects (post-condition)

Notation:
“\{\text{pre-condition}\}\text{action}\{\text{post-condition}\}”
Group and Homomorphism [DF04]

- **Group:**
  - Set of mathematical objects with an intern operation $\circ$.
  - Properties:
    - $\circ$ is associative,
    - There is a unique neutral element for $\circ$,
    - Each element has an unique inverse.

- **Group Homomorphism:**
  - Mathematical function from one group to another preserving the group structure: $F(r_1 \circ r_2) = F(r_1) \circ F(r_2)$. 
1 Introduction
   ■ Context
   ■ Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   ■ Future Work
   ■ Conclusion

6 Bibliography
Vocabulary: What is a transformation?

Transformation:
- Mathematical and reifiable object.
- Modification of a model (software representation).
- Defined under a meta-model (model specification [BP01]).

Notation $\mathbb{T}_{MM}$: set of all transformations under the meta-model $MM$.

Our goal: Transpose them between meta-models.

Example: Renaming a class could be a transformation.
Transformations as Hoare Triple

Transformation = Modification:
- Is an action,
- Depends on a previous model state,
- Creates a new model state

Example:

\{\exists C_0 \land \#C_1\} \text{Rename } C_0 \text{ in } C_1\{\#C_0 \land \exists C_1\}
Set of Transformations as Groups

- is sequencement ("followed by", "then").
  - Internal Operation:
    \[ \forall a, b \in T_{MM}, a \circ b \in T_{MM} \]
  - Associativity:
    \[ \forall a, b, c \in T_{MM}, (a \circ b) \circ c = a \circ (b \circ c) \]
  - Unique Neutral Element (Identity):
    \[ \exists ! \mathbb{1} \in T_{MM} \text{ st } \forall a \in T_{MM}, a \circ \mathbb{1} = \mathbb{1} \circ a = a \]
  - Unique Inverse:
    \[ \forall a \in T_{MM}, \exists ! a^{-1} \in T_{MM} \text{ st } a \circ a^{-1} = a^{-1} \circ a = \mathbb{1} \]
  - Inversion of sequence:
    \[ \forall a, b \in T_{MM}, (a \circ b)^{-1} = b^{-1} \circ a^{-1} \]

Example: The reverse of Renaming a class *FOO* as *BAR*, is to rename the class *BAR* as *FOO*. 
Transpositions as Group Homomorphism

Transformations are elements of Groups, therefore
Transpositions are Group Homomorphisms.

\[ F(a \circ b) = F(a) \circ F(b) \]
\[ F(\mathbb{1}_{\text{MM1}}) = \mathbb{1}_{\text{MM2}} \]
\[ F(a^{-1}) = F(a)^{-1} \]

Example: There would be a transposition between the code, the developer work on, and the model, the analyst work on.
Commutativity

commuting transformations = change modifications order.

Example: “Renaming FOO in BAR, then Adding a method baz in BAR” becomes “Adding a method baz in FOO, then Renaming FOO in BAR”.
1 Introduction
   ■ Context
   ■ Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   ■ Future Work
   ■ Conclusion

6 Bibliography
Implementation

- PADL [Gué03, AA03, GA08]
  - Meta-model used to represent specification of programs,
  - High-level models.
  - Developed to represent patterns and abstract designs,

- JCT
  - Meta-model used to represent program code source,
  - Low-level models (Bound Abstract Syntax Tree),
  - Developed to represent Java Program, similarly to javac.
1 Introduction
   ■ Context
   ■ Running Example

2 Background

3 Transformations: A Mathematical Formalism

4 Implementation

5 Future Work and Conclusion
   ■ Future Work
   ■ Conclusion

6 Bibliography
Future Work

Implementation in progress:

- JCT implementation almost finalized,
- PADL and JCT transformations implementation in progress,
- Transposition between JCT and PADL to specify and implement,
- PADL and JCT transformations commutativity implementation in progress.
Conclusion

Our approach provides:

- Mathematical theory, verifiable, formal.
- Live feedback and concurrent modifications of the program, at many levels of abstraction.
- Reversibility (Rollback facilities).
- Traceability of each transformations.

But is purely theoretical now. Implementation in progress.


Bibliography II


The End!

Thank You!

Question?