Automated traceability analysis for UML model refinements

Lionel C. Briand, Yvan Labiche, Tao Yue
Information and Software Technology (2009)

Presented by
AhRim Han

2009. 2. 4
Contents

- Introduction
- Related work
- Overall approach
- Traceability analysis
- Case study
- Tool support
- Conclusion & Future work
UML-based iterative software development

- Stepwise refinement of analysis (abstract) model into design (refined) model
  - They are at the different levels of abstraction
- Two models must remain consistent as they are incrementally refined and changed
Impact analysis

- Definition
  - Identify the potential consequences of a change, or estimating what needs to be modified to accomplish change

- Horizontal vs. vertical impact analysis
  - Both require traceability between models elements
In this paper, we propose an approach to support

- **Traceability analysis** for vertical impact analysis (VIA) of UML class diagrams in a (semi-) automated way
  - Establish traceability links based on the identified refinements, during vertical evolution
    - Identify atomic changes
    - Derive refinements from the identified atomic changes, thus capturing the user’s intent at a higher level of abstraction than atomic changes

- **Refinement:**
  - A transformation that adds more detail to an existing models
  - Also called realization
  - In this paper, we treat refactoring and change design pattern as special type of refinement
Related work (1/2)

- Refinements classification
  - Egyed’s work (2004)
    - Only take into account a few refinements which are not organized in a systematic way
      - Refinement can not be identified automatically

- Refinements identification
  - Shen’s work (2002), Mens’s work (2000)
    - Request stakeholders to explicitly specify refinements by means of stereotypes
      - Significant overhead on the user’s part
Automatic traceability link creation

- Letelier’s work (2003)
  - Require stakeholders to define traceability links between UML model elements
    - Do not indicate which kind of traceability information should be captured during the refinement process
    - User’s intent can not be captured at the higher level of abstraction
Overall approach

- Automatic VIA of UML models

Scope of this paper:

1. Identify atomic changes
2. Identify refinements from identified atomic changes
3. Establish traceability links between model elements of UML model versions
4. Perform vertical impact analysis (VIA) between abstract model and refined model
• **Traceability Link**: A relationship that describes the traceability connection between a model element of one UML model and a model element of another UML model
Taxonomy of atomic changes

- 47 concrete atomic changes for a class diagram

- Atomic changes can be detected using the compare & merge engine which is available in Rational Software Architect (RSA)
Taxonomy of refinements

- 31 concrete atomic refinements for a class diagram
Refinement specification

For each concrete refinement, the template is provided containing the following information:

- A general description
- User’s intention
- A list of atomic changes
- User help if required
- Constraints the atomic changes must satisfy (w/OCL)
- A description of the corresponding traceability links (w/OCL)

### C.1.1 TopDownGen

**Description:** Introduce subclasses to the class being refined using generalization to suggest general-specific relationships.

**User’s Intention:** Detail the responsibility of the original class by introducing subclasses using generalization, because these subclasses, though similar, have significant differences in the interface or services they provide.

**Atomic Changes:** Two kinds of atomic changes are required to derive the refinement: AddedClass and AddedGeneralization. Note that we only account for the situation in which only one subclass is added. If more than one subclass is added, more than one occurrence of the refinement of type TopDownGen is identified. The specification of refinement TopDownGen, i.e., the characterization of its atomic changes and their relations, is shown as the OCL invariant for class TopDownGen in Figure 30 (a).

**Traceability Links:** Two traceability links have to be established: one between the original class and the superclass of the added generalization; one between the original class and the added subclass. These traceability links are specified in the OCL expression in Figure 30 (b).

**User Help:** none.

---

(a) Constraint on atomic changes

```
Context: TopDownGen
self.atomicChanges->size() = 2
and self.atomicChanges->exists(oclIsTypeOf(AddedClass))
and self.atomicChanges->exists(oclIsTypeOf(AddedGeneralization))
and self.traceabilityLinks->select(oclIsTypeOf(AddedGeneralization)).affectedElement.
oclAsType(Generalization).specific->includes(self.atomicChanges->
select(oclIsTypeOf(AddedClass))).affectedElement.oclAsType(Class))
```

(b) Traceability links

```
Context: TopDownGen
let newGen = self.atomicChanges->select(oclIsTypeOf(AddedGeneralization)).
.affectedElement.oclAsType(Generalization)
let superClass = newGen.general
let subClass = newGen.specific
let origClass = self.originalModel.diagram->select(oclIsTypeOf(ClassDiagram)).
.modelElements->select(oclIsTypeOf(Class))--select(name = superClass.name)
in
self.traceabilityLinks->size() = 2
and self.traceabilityLink->exists(origin = origClass and target = superClass)
and self.traceabilityLink->exists(origin = origClass and target = subClass)
```
Traceability link establishment

- After identifying the refinement, a set of traceability links are established (semi-) automatically
- Example
  - Atomic refinement ‘TopDown’
Subject system: Arena

- Non-trivial, text book case study for illustrating object-oriented software development
  - Distributed, multi-user system for organizing and conducting tournaments

- Two different abstraction level class diagrams
  - Analysis class diagram is obtained from the book
  - Design class diagram is reversed from the code

Class diagrams characteristics

<table>
<thead>
<tr>
<th></th>
<th>Class diagram (analysis)</th>
<th>Class diagram (design)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Classes</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td># Interfaces</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td># Associations</td>
<td>17</td>
<td>34</td>
</tr>
<tr>
<td># Dependencies</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td># Generalizations</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td># Model elements</td>
<td>41</td>
<td>68</td>
</tr>
</tbody>
</table>
Goal

- Completeness of the atomic changes
  - To assess whether our atomic change taxonomy was complete with respect to our case study
- Completeness and correctness of refinement rules
  - To determine whether our refinement rules could account for all atomic changes
  - To determine whether they led to correct deductions regarding the refinements made to the analysis model to obtain the design model
Case study (3/4)

- Validation procedure and results

- 90 atomic changes, from 12 different atomic changes

- Taxonomy of atomic changes was sufficient to explain how the analysis model can be transformed into the design model

- 66 refinements, from 18 different refinements were necessary to complete the class diagram transformation from analysis to design

- 66 identified refinements accommodate all the 90 atomic changes.

- Identified refinements were able to explain why and how the Arena analysis model was refined into the design model, while conforming to the information provided in the text book
Result summary

- Taxonomies of atomic changes and the set of refinement rules we proposed are complete and correct based on the Arena case study.
Tool support

- VIATool (Vertical Impact Analysis Tool) architecture
  - Built as a set of java plug-ins using IBM RSA
  - Architected by integrating following technologies
    - Eclipse development platform, EMF, OCL engine, CompareMerge engine, and Eclipse UML
  - Easily extended and accommodate certain changes
Conclusion & future work

❖ Conclusion

- Provide automated traceability analysis between models at different levels of abstraction
  - Classification of atomic changes and refinements
    - Capturing the changes’ intent at a higher level of abstraction than model element changes
  - Automated identification of refinements based on detected changes
  - Automated establishment of traceability links

❖ Future work

- Extends the work to the complete set of UML diagrams
- Complete the methodology of performing vertical impact analysis
  - Different kinds of links have different impacts
Thank You.
• **Refinement**: Abstract (original) model → Concrete (refined) model
• **Traceability Link**: A relationship that describes the traceability connection between a model element of one UML model and a model element of another UML model
Figure 107 ReplaceSuperclassWithInterface - Specification
Case study: Arena

Summary of refinements and atomic changes

<table>
<thead>
<tr>
<th>Refinement type (total 18)</th>
<th>No. of occurrences</th>
<th>Changes made to the analysis mode</th>
<th>Atomic Changes (for one instance of a refinement type) with no. of occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Part A): Summary of refinements and atomic changes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| CollapseHierarchy         | 2                  | Classes User and Player -> class User  
classes User and LeagueOwner -> class User | DeletedClass (1)  
DeletedGeneralization (1)  
DeletedClass (1)  
DeletedInterface (1)  
DeletedGeneralization (1)  
AddedDependency (1)  
DeletedAssociation (1)  
AddedClass (1)  
AddedAssociation (3)  
AddedClass (1)  
AddedGeneralization (1) |
| ReplaceSuperclassWithInterface | 2              | Superclass Game -> interface Game  
Superclass TournamentStyle -> interface TournamentStyle |                                                                                |
| AddedBridgeClass          | 1                  | Association between Tournament and Match -> new class Round and associations |                                                                                |
| TopDownCon                | 2                  | Class Match -> classes Match and TicTacMatch |                                                                                |
| InlineClass               | 1                  | Class Round -> classes Round and KnockOutRound  
Classes Interest Group and User -> class User | DeletedClass (1)  
DeletedAssociation (1)  
AddedClass (1)  
AddedAssociation (3)  
AddedClass (1)  
AddedDependency (1) |
| DetailAssoFunctionality   | 1                  | Association between Arena and Game -> path between Arena and Game |                                                                                |
| ByBridgeClass             | 1                  | Association between Arena and Interest Group -> path between Arena and User via bridge class GateKeeper |                                                                                |
|                          |                    | Association between Arena and LeagueOwner -> path between Arena and User via bridge class League (rename owner) |                                                                                |
| ClassIsAbstractRef        | 2                  | Class User: abstract -> concrete  
Class Match: concrete -> abstract | ChangedClassIsAbstract (1) |
Case study: Arena

Two example of refinements

- **DetailAssoFunctionality**: Refine an association by adding new classes and association

- **ReplaceSuperclassWithInterface**: Replace a superclass with an interface and replace the general with implementation relationship
Case study: Arena

Identified refinement in the case study

- New class Round corresponds to a set of Matches that can be held concurrently (page 381)

Actual intention referred in the book

- New class Round “corresponds to a set of Matches that can be held concurrently” (page 381)

Result

- This conforms to the intent of our identified refinement

AddBridgeClass

- The association between classes Tournament and Match in the analysis cannot model the requirement of organizing a set of matches held concurrently
- A new bridge class Round has to be added to bridge classes Tournament and Match