UML Model Refactoring

Viktor Stojkovski

University of Antwerpen, Faculty of Computer Science, Masters of Software Engineering
January, 2013, Belgium

Email: Viktor.Stojkovski@student.ua.ac.be

Introduction to UML Model Refactoring

- Because of constant evolution, systems must be modified and evolve
- There is no precise validation whether the modifications won’t change the system behavior
- Necessity of basic set of model transformations (refactorings)

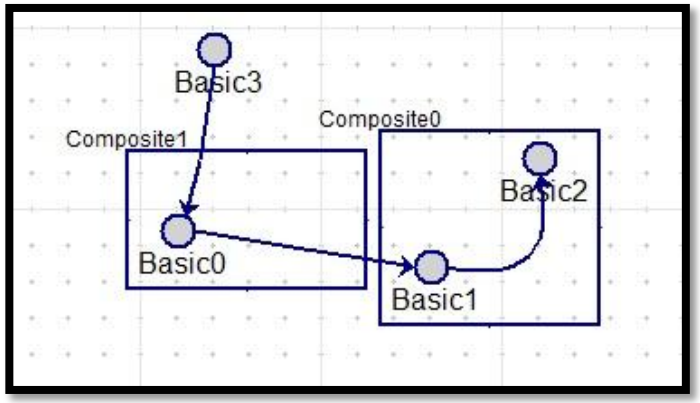
Main goal of the project

- Create UML Statechart refactoring rules using the defined transformations in the article Refactoring UML Models
- Define GraphGrammar rules using the Meta-Modeling software AToM^3

- Sunye, G., Pollet, D., Traon, Y. L., Jezequel, J.-M., 2001. Refactoring UML Models. Springer.
- http://atom3.cs.mcgill.ca/

Working process(1)

1. Reading and understanding the defined rules and the OCL (Object Constraint Language) constraints
2. Create a testing model in AToM^3 using the DCharts as formalism

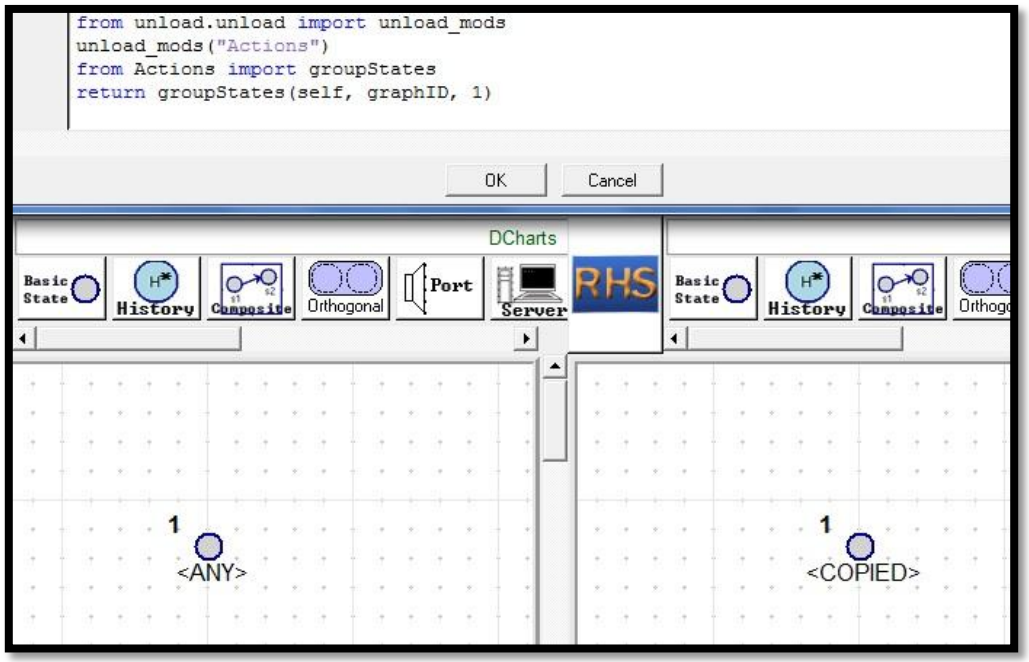


Group States model example

Feng, H., 2004. Dcharts, a formalism for modeling and simulation based design of reactive software systems

Working process(2)

3. Specify the grammar by creating the rules – LHS and RHS



Group States rule example

Working process(3)

4. Add conditions to the rules as pre-conditions and actions as post-conditions
5. Program the pre and post conditions in Python

```
foldIncomingActionsCond(self, graphID, stateLabel):
state = self.getMatched(graphID, self.LHS.nodeWithLabel(stateLabel))
enterAction = state.enter_action.toString().strip()
# compare if all the input transitions actions are the same
allInTransActSame = compareTransAct(state, "in")
# get the number of incoming transitions, their number should be larger that
nrInTrans = countTrans(state, "in")
empTrans = emptyTransition(state, "in")
# the condition will be satisfied if the state has no entry action, all the
# are the same, the number of transitions is larger than 1 and the actions a
# empty fields
if not enterAction and allInTransActSame and nrInTrans > 1 and not empTrans:
    return 1
else:
    return 0
```

Precondition for the Fold Incoming Actions rule

```
foldIncomingActions(self, graphID, stateLabel):
state = self.getMatched(graphID, self.LHS.nodeWithLabe
action = getActFromTrans(state, "in")
transitions = []
state.enter_action.setValue(action)

for trans in state.in_connections_:
    if isinstance(trans, Hyperedge):
        for checkState in trans.in_connections_:
            if isinstance(checkState, Basic):
                transitions.append(trans)
deleteActFromTrans(transitions)
```

Acton for the FIA rule

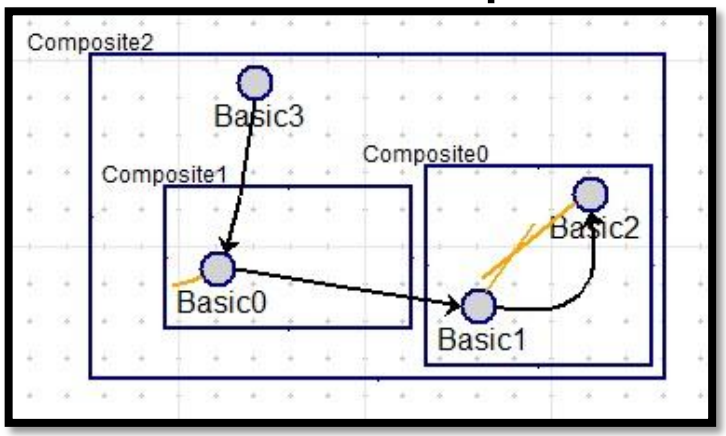
Working process(4)

6. Organizing the rules in the grammar by giving them priorities
   – The order of the rules is very important (explanation!?)

Hierarchy of the rules in the GG

Working process(5)

7. Executing the grammar
8. Analyzing the results:
   - Does the execution of the grammar over the statechart give the expected results?
   - Is the model behavior preserved?



Example of the Group State rule execution

Presentation of an example from the practical work of the project

- Refactoring a Statechart diagram that is modeling a phone call
- Explanation of the refactoring process through an examples
- Comparing the results of the refactoring to the theoretical results from the article

Sunye, G., Pollet, D., Traon, Y. L., Jezequel, J.-M., 2001. Refactoring UML Models

Conclusion

- After testing the refactoring grammar on a number of UML Statechart models the results were satisfactory
- Room for improvements:
  - More wide-ranging pre and post conditions covering all of the states in which the model can be
  - Inventing new and expanding the already specified refactoring rules (example with the extension of the Move State out of Composite rule)

References

- Feng, H., 2004. Dcharts, a formalism for modeling and simulation based design of reactive software systems. A Masters Thesis
- OMG, 2009. OMG Object Constraint Language (OCL). OMG
- Selic, B., 2009. Unified Modeling Language Specification (version 2.1). OMG
- Sunye, G., Pollet, D., Traon, Y. L., Jezequel, J.-M., 2001. Refactoring UML Models. Springer
- Yang, M., Michaelson, G. J., Pooley, R. J., 2008. Formal action semantics for a UML action language. Journal of Universal Computer Science
- http://atom3.cs.mcgill.ca/

Questions & Discussion