Incremental Test Case Generation for UML-RT Models

Eric James Rapos, Juergen Dingel
{eric,dingel}@cs.queensu.ca
Modeling & Analysis in Software Engineering Group
School of Computing
Queen's University, Kingston, Ontario, Canada

Background

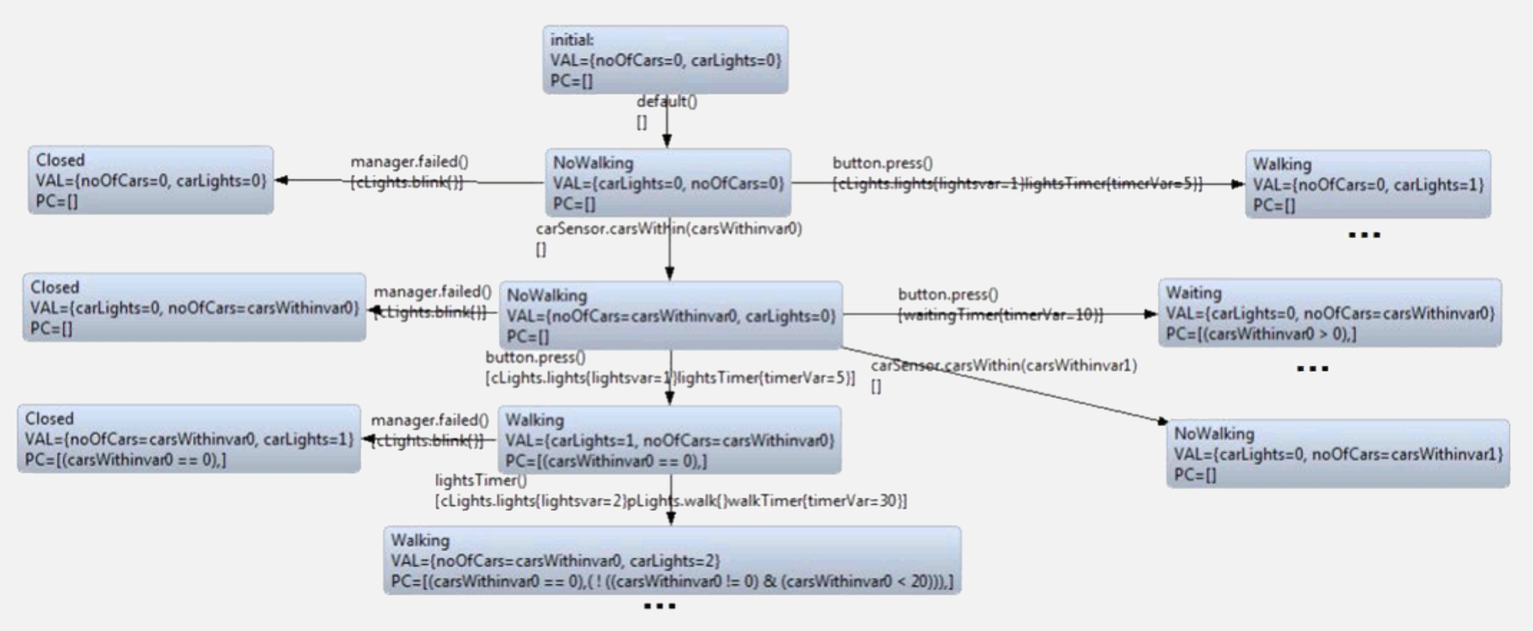
Model Driven Development

- Incremental Process (M1→ M2 → … Mn … → Generated Code)
- More and more commonly used, especially in real-time systems

Symbolic Execution

- Model program behaviour
- Existing work allows generation of Symbolic Execution Tree (SET)
- Using SETs, automatic test case generation can occur
- SETs are useful in analyzing program changes

Example SET



Motivation

Furthering of Research in Model Driven Development

- Improve usability of MDD techniques
- Develop tools for developers
- Work on cutting edge research

Improve Efficiency of Test Case Generation

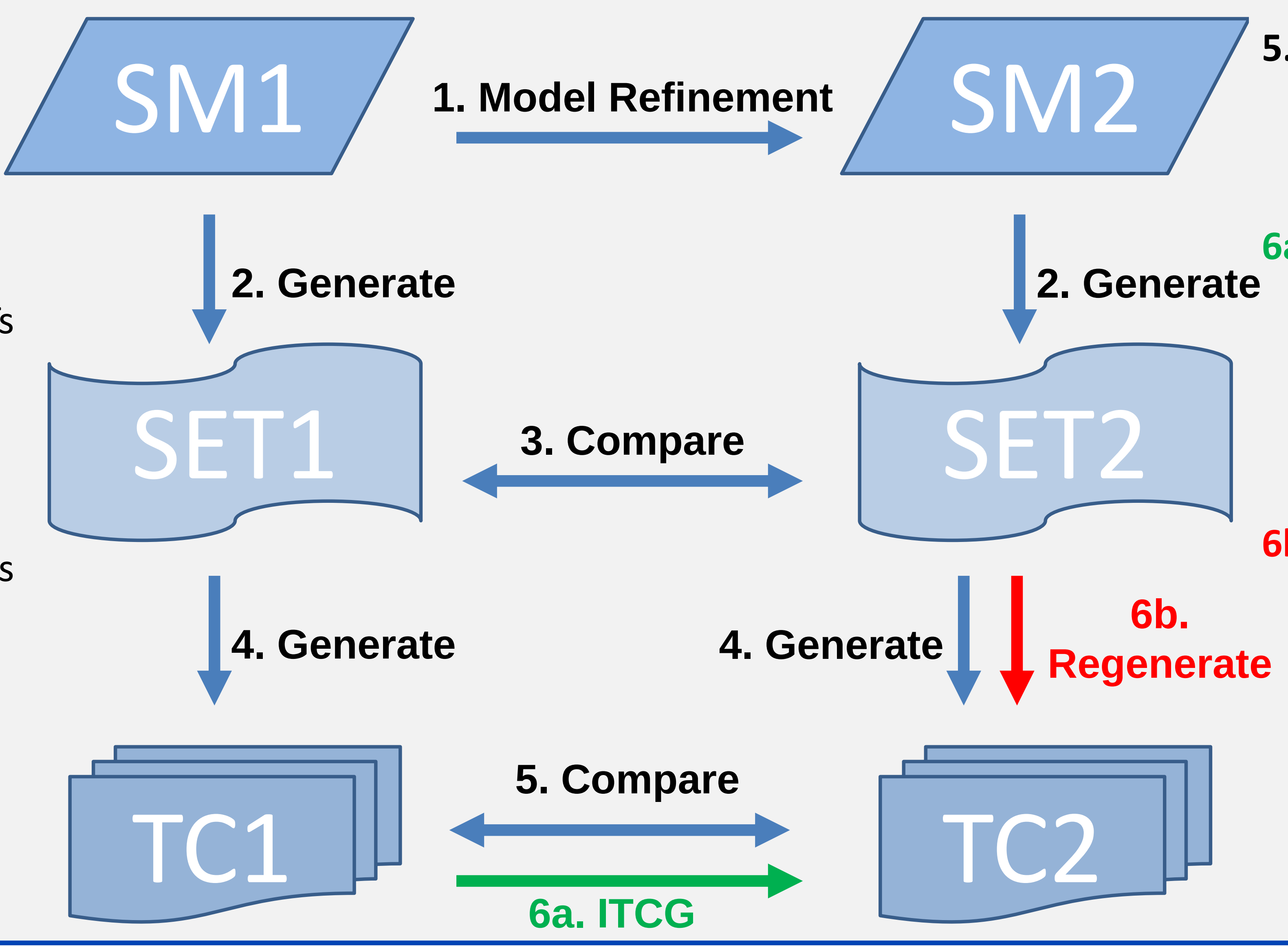
- Automatic regeneration of test cases can be inefficient and sometimes redundant
- Make only the necessary changes to a test case
- Use an incremental process, to coincide with the MDD process

Understand Effects of Model Transformations

- Each type of change to model will have certain effects on the SET and test cases
- We hope to categorize all typical model evolution steps in order to understand how they effect the artifacts of MDD

The Process

1. **Model Refinement**
   - Perform changes to models
   - Use collected set of changes
2. **Symbolic Execution Tree Generation**
   - Existing process
   - Automated generation of SETs
3. **Comparing Effects on Symbolic Execution**
   - Determine effects of changes on SETs
   - Create rules based on findings
4. **Automatic Test Case Generation**
   - Existing process
   - Use SETs to automate generation of test cases
   - Initially for both models, for comparison, but only on original model in the end



5. **Comparing Effects on Test Cases**
   - Determine effects of model changes on test cases
   - add to rules from Step 3

6a. **Incremental Test Case Generation**
   - Goal of our work
   - Able to make changes to model and incrementally generate test cases based on rules

6b. **Regeneration of Test Cases**
   - We want to avoid this at all costs
   - Same process as Step 4
   - This is what we feel can become inefficient and redundant
   - Used solely as a last resort.

Planned Work

Develop ECORE Model of Symbolic Execution Trees

- Standard representation of Symbolic Execution Trees
- Output of Symbolic Analysis
- Used for Comparison in Step 3 from above

Collect a Standard Set of Model Evolution Steps to Evaluate

- Begin with Bran Selic's paper on refinement patterns [Sel11]
- Use four categories:
  - No change
  - Renamings
  - Additions
  - Deletions

Generate Test Cases and Compare Differences

- Use the collected set to compare different model changes on test cases

Develop a Functioning Prototype That Will Automate The Process

- Automate the process carried out above in a software prototype

Expected Outcomes

A Set of Rules on Model Evolution

- For each standard model evolution step, determine its effect on:
  - Symbolic Execution Tree
  - Test Cases
- Investigate non-standard evolution as well to determine possible effects
- Formulate a set of rules based on findings

Better Understanding of State Machine Evolution

- The above rules will not only be useful in our work, but as a better understanding of the MDD Process

A Software Implementation

- **Input to tool:** original model, test case for original model, and the evolved model
- **Functionality:** Use "The Process" to determine effects on test case
- **Output from tool:** modified test case for evolved model
- **Future:** Potential for integration with development environment

Resources

1. [Sel11] Bran V. Selic, "A Short Catalogue of Abstraction Patterns for Model-Based Software Engineering", to appear in Journal of Software and Informatics (2011)
2. [ZD11a] K Zurowska, J Dingel, "Symbolic Execution of UML-RT State Machines", DRAFT (2011)
3. [ZD11b] K Zurowska, J Dingel, "Modular Symbolic Execution of Communicating and Hierarchically Composed UML-RT State Machines", DRAFT (2011)
4. [UKB10] Engin Uzuncaova, Sarfraz Khurshid, Don S. Batory, "Incremental Test Generation for Software Product Lines", IEEE Trans. Software Eng. 36(3): 309-322 (2010)
5. IBM Rational Software Architect Real-Time Edition (RSA-RTE) - http://www-947.ibm.com/support/entry/portal/Overview/Software/Rational/Rational_Software_Architect_RealTime_Edition
6. Eclipse Modeling Framework (EMF) - http://www.eclipse.org/modeling/emf/