Incremental Test Case Generation for UML-RT Models

Eric J. Rapos, Juergen Dingel, James R. Cordy
Modeling & Analysis in Software Engineering Group
Software Technology Lab
School of Computing, Queen's University
Kingston, ON
{eric, dingel,cordy} @cs.queensu.ca

Introduction & Motivation

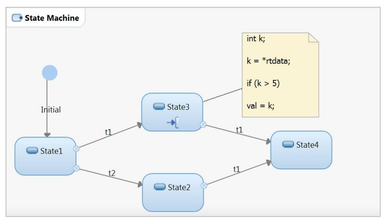
The iterative nature of model-driven engineering (MDE) gives rise to redundant test case regeneration

Our goal was to understand and classify the effects of model evolution on test cases

The overall aim was to improve the efficiency of test case maintenance tools by reducing this redundancy

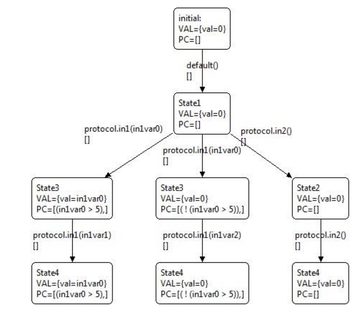
Background

UML-RT



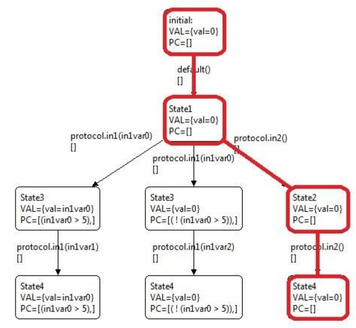
- Real-time behavioural modeling of systems
- States, transitions, triggers, and actions
- RSA-RTE implementation

Symbolic Execution



- All possible executions of a system are computed
- Symbolic values are used

Test Case Generation



- A Test Case is generated for each path through the tree
- Automated process

Model Change Classifications

1. Determine 14 *Standard Evolution Steps*
2. Perform these steps on 5 different models
3. Compare resulting Symbolic Execution Trees (SETs) with original
4. Determine the impact of evolution on execution (sectioned, uniform, or untraceable)
5. Determine what type of update needed to SET (partial symbolic execution, direct update, or full symbolic execution)

Evolution	Impact on SET	Action Required
1. Modify State	Uniform	Direct Update
2. Delete State	Sectioned	Direct Update
3. Delete Transition	Sectioned	Direct Update
4. Add Parameter	Uniform	Direct Update
5. Add Transition	Sectioned	Partial Symbolic Execution
6. Add State	Sectioned	Partial Symbolic Execution
7. Modify Transition	Sectioned	Partial Symbolic Execution
8. Add Entry Code	Sectioned	Partial Symbolic Execution
9. Modify Entry Code	Sectioned	Partial Symbolic Execution
10. Delete Entry Code	Sectioned	Partial Symbolic Execution
11. Add Action Code (output signal)	Sectioned	Partial Symbolic Execution
12. Add Action Code (value change)	Sectioned	Partial Symbolic Execution
13. Delete Parameter	Uniform	Partial Symbolic Execution
14. Modify Initial Value	Untraceable	Full Symbolic Execution

Incremental Test Case Generation

Model Differencing
Use RSA-RTE's internal differencing tool to identify differences between original and evolved model.
Inputs: 2 RSA-RTE Models **Outputs:** List of Differences between Models

Choose Best Action
Based on difference, select best option.

1. **Direct Updates to SET and Test Suite**
Directly update the SET and test suite using information available from differencing models
Inputs: SET and Tests
Outputs: Updated SET and Tests

2. **Partial Symbolic Execution**
When not possible to directly update, second preference is partial symbolic execution.
Inputs: Old SET and new model
Outputs: New SET

3. **Full Symbolic Execution**
As a last resort, we may need to fully symbolically execute the entire model.
Inputs: new Model
Outputs: New SET

Test Case Generation
A full test suite is generated for the newly obtained SET.
Inputs: New SET **Outputs:** Generated Test Suite

Test Suite Differencing
Given the new test suite, we difference it with the original test suite and determine which of the test cases have changed or been added or removed, so we know which tests need to be run, and which do not.
Inputs: Original and New Test Suites **Outputs:** List of added and/or removed tests

Validation

Procedure

1. Generate changed model versions
2. Symbolically execute each changed model, and generate tests using existing methods
3. Produce new test suite using our tool
4. Compare execution times of two versions

Results

	Model 1	Model 2	Model 3	Model 4	Model 5
1. Direct Updates					
1	55.72%	36.90%	34.95%	15.59%	55.55%
2	26.07%	26.40%	30.69%	5.65%	11.22%
3	34.13%	22.48%	42.13%	31.30%	33.10%
4	56.05%	19.66%	34.56%	39.81%	38.93%
2. Partial Symbolic Execution					
5	31.63%	-26.20%	-3.44%	-1479.38%	1.42%
6	26.74%	-13.66%	13.17%	-476.50%	-5.46%
7	23.29%	-21.30%	-21.78%	-206.29%	5.31%
8	28.62%	-45.20%	3.92%	-21.61%	-9.03%
9	34.21%	-46.31%	5.44%	-5.60%	-5.70%
10	-35.14%	-30.15%	-124.66%	-13.65%	26.42%
11	33.82%	4.28%	-251.68%	-5.50%	-88.72%
12	32.45%	-18.39%	-951.18%	-443.95%	5.78%
13	-249.64%	-45.39%	-65.91%	-60.46%	-19.64%
3. Full Symbolic Execution					
14	-2.89%	-38.40%	-53.12%	-22.00%	-25.85%

average performance of our tool (IncreTesCaGen) over five models, compared to traditional test generation methods (percentage of gain/loss in time)

Conclusions

Direct Update evolutions performed the best, requiring no symbolic execution or test generation

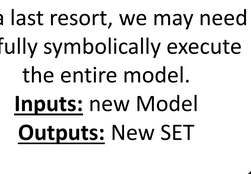
Models which generate SETs containing subsumption generally performed poorly

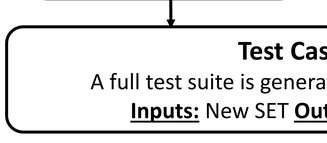
Larger models showed more of a performance improvement than smaller models

The location of the change in the model impacted performance

Future Work

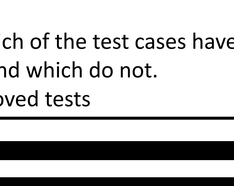
Shift focus to other formats, including Simulink Models





Heavier focus on the maintenance of production tests through co-evolution

Examine real-world models to demonstrate the industrial merits of our work



References

1. E.J. Rapos. "Understanding the Effects of Model Evolution through Incremental Test Case Generation for UML-RT Models". MSc Thesis. Kingston, Canada, September 2012.
2. E.J. Rapos and J. Dingel. "Incremental Test Case Generation for UML-RT Models Using Symbolic Execution". Poster at IEEE International Conference on Software Testing, Verification and Validation (ICST'12). Montreal, Canada, April 2012.
3. K. Zurowska and J. Dingel. "Symbolic Execution of UML-RT State Machines". 27th ACM Symposium on Applied Computing, Track on Software Verification and Testing (SAC-SVT'12). Riva del Garda, Italy, March 25-29, 2012.
4. K. Zurowska and J. Dingel. "SAUML - a Tool for Symbolic Analysis of UML-RT Models". Tool Demonstration Paper. 26th IEEE/ACM International Conference On Automated Software Engineering (ASE'11). Lawrence, Kansas, USA, Nov 6-10, 2011
5. K. Zurowska and J. Dingel. "Symbolic Execution of UML-RT State Machines". Technical Report 2011-578, School of Computing, Queen's University, June, 2011.
6. B. Selic. "Using UML for Modeling Complex Real-Time Systems". Muller, F., Bestavros, A. (eds.) LCTES 1998. LNCS, vol. 1474, pp. 250–260. Springer, Heidelberg (1998)
7. N.H. Lee, S.D. Cha. "Generating test sequence using symbolic execution for event driven real-time systems". Microproc. and Microsys. 27, 523–531 (2003)
8. IBM Rational Software Architect RealTime Edition - http://www-947.ibm.com/support/entry/portal/overview//software/rational/rational_software_architect_realtime_edition
9. EMF Compare - http://www.eclipse.org/emf/compare/