Using Model-Based Testing for Testing Application Models in the Context of Domain-Specific Modelling

Olli-Pekka Puolitaival
Index

• Domain-Specific Modeling

• Where the Bug Lures?

• How to Find Bugs Out?

• Laboratory example: Coffee machine

• Conclusion
Domain-Specific Modeling
Where the Bug Lures?

1. Application model:
   - Specification → Application model

2. Code generator:
   - Model → Code generator → Code

3. Framework:
Testing in the Context of DSM

- Iterative and incremental language development
  - Code generator produces same fault many places and therefore those are easy to find out
- Problems
  - Ad-hoc testing
  - Test coverage?
  - Test maintenance?
- More systematic and automated testing is required
Model-Based Testing (MBT)

- Model-Based Testing (MBT) is about generating test cases from a model
- Idea:
  1. Make a high level behaviour model of system under testing
  2. MBT tool generates test suite automatically
  3. Run tests in your test execution environment
- Benefits of MBT
  - Reduces maintenance effort
  - Increases test coverage
Testing Presuppositions

= Fact

= Testing / Verification

= System Under Testing
How to Find Bugs?
Case study:

COFFEE MACHINE
Demonstration Structure

• Architecture of the coffee machine:
  • User interface = User interface hardware
  • Control logic = Control logic code
  • Machine hardware = Make coffee in practice: add water, warm water, add coffee….

• DSM is used for modelling control logic part
• Also other parts are generated for demonstration
Demo Setup

**MetaEdit+**
- Application models
- Python generator
- Mod2QML
  - Python code

**Confromiq Qtronic**
- Model in Qtronic’s format
- QML Model
  - Qtronic test generation
- Test suite

**Virtual Coffee Machine**

**Testing**

**Test executor**
DSML for coffee machine
Model transformation

- MetaEdit+
  - Application models
  - Python generator
  - Mod2QML
  - Python code

- Confromiq Qtronic
  - Model in Qtronic’s format
  - QML Model
  - Qtronic test generation

- Test suite

- Test executor

Virtual Coffee Machine
Model transformation
Generated test cases
Demonstration summary

- MetaEdit+
  - Application models
  - Python generator
  - Mod2QML
- Confromiq
  - Model in Confromiq's format
- QML Model
  - Qtronic test generation
- Test suite
- Testing
- Test executor
- Virtual Coffee Machine
Future Research: Passing model transformation
Future Research

• Language testing using application generation (see my presentation in OOPSLA 2008 DSL workshop)

• Adopting DSM in testing area:
  • Domains specific model-based testing
  • Test case visualization using DSM
Thank you!