Model-Based Testing

TestNet  Werkgroep  MBT
Software Testing

Checking or measuring some quality characteristics of an executing software object by performing experiments in a controlled way w.r.t. a specification.

SUT

System Under Test

tester

specification
Testing: Why Something New?

• Quest for improved quality, with less time and cost
  — detect more bugs faster and cheaper

• Changing processes:
  — scrum, agile
  — model-driven

• Increasing system size, complexity

• Blurring boundaries
  — time: continuous change, continuous delivery
    • deliver fast, deliver often  →  test fast, test often
  — space: complex interactions, systems-of-systems
    • end-to-end testing  →  where is the end?

Software bugs cost US economy $59,500,000,000 (www.nist.gov)
Testing Complexity

Testing effort grows exponentially with system size

Testing cannot keep pace with development

- $x: [0..9] \rightarrow 10 \text{ ways that it can go wrong}$
  $\rightarrow 10 \text{ combinations of inputs to check}$

- $x: [0..9] \rightarrow 100 \text{ ways that it can go wrong}$
  $\rightarrow 100 \text{ combinations of inputs to check}$

- $y: [0..9] \rightarrow 1000 \text{ ways that it can go wrong}$
  $\rightarrow 1000 \text{ combinations of inputs to check}$

\(\rightarrow\) combinatorial explosion of required testing effort
Testing Complexity

Testing effort grows exponentially with system size

Testing cannot keep pace with development

→ combinatorial explosion of required testing effort
MBT

model-based test generation

test execution

system

model

pass fail
What is MBT?

- Model-Based Testing
- Next step in test automation
  + automatic test generation and result analysis
- New way of testing to detect more bugs faster and cheaper
- Testing with powerful tools
- Testing based on an abstract system **model**
  that specifies the behaviour of (an aspect of) the SUT
MBT: Next Step Test Automation

- Model-Based Testing
- Keyword-Driven
- Scripted
- Record & Playback
- Manual Testing
1. Manual testing

Testing 1: Manual Testing
Testing 2 : Scripted Testing

1. Manual testing
2. Scripted testing

- Manual testing:
  - Execute test cases
  - SUT
  - Pass/Fail

- Scripted testing:
  - Execute test cases
  - SUT
  - Pass/Fail
Testing 3: Keyword-Driven Testing

1. Manual testing
2. Scripted testing
3. Keyword-driven testing
Testing 4: Model-Based Testing

1. Manual testing
2. Scripted testing
3. Keyword-driven testing
4. Model-based testing
MBT Improves the Testing Process

- Automatic testing
  - *test generation + execution + analysis*

- Detecting more bugs faster and cheaper
  - *more, longer, and more diversified tests*

- Model is precise and consistent test basis
  - *unambiguous analysis of test results*

- Early error detection during model construction and analysis
  - *improved understanding of system requirements*

- Test maintenance by maintaining models
  - *improved regression testing*
And Moreover . . . MBT . . .

- Enabling **flexibility** in modelling alternative, concurrent, distributed, exceptional, uncertain, . . . behaviours
- **Compositionality** : building large models from simple ones
- Expressing **test coverage** by model coverage
- **Diagnosis** through model analysis
- Linking to **model-based system development**
- Promoting **uniformity** of tests and test processes
MBT: Abstraction + Composition

- Four tasks in parallel, in any order

\[
\text{taskA} := \text{task}(\text{startA?}, \text{readyA!}) \\
\text{taskB} := \text{task}(\text{startB?}, \text{readyB!}) \\
\text{taskC} := \text{task}(\text{startC?}, \text{readyC!}) \\
\text{taskD} := \text{task}(\text{startD?}, \text{readyD!})
\]

\[
\text{model} := \text{taskA} || | | | \text{taskB} || | | | \text{taskC} || | | | \text{taskD}
\]
MBT: Abstraction + Composition
Doing Something with Models

- **Modelling**  
  making a model reveals errors

- **Simulation**  
  go step-by-step through the model

- **Model checking**  
  go through all states of the model

- **Theorem proving**  
  prove theorems about the model

- **Code generation**  
  executable code from the model

- **Testing**  
  test an implementation for compliance

- **Model learning**  
  generate a model from observation
Validation, Verification, and Testing

- Informal requirements
- Validation
- Model
- Verification
- (Model-based) testing
- SUT
- Real world
A model is more \textit{less} than code generation:

- views
- abstraction
- testing of aspects
- verification and validation of aspects
Code Generation from a Model?

- specification of properties rather than construction
- under-specification
- non-determinism

$y \times y = x$

model of $\sqrt{x}$
MBT : Status & Challenges

Promising, emerging approach, but

• Who does the modelling?
  extra effort of making models compensated by better tests

• Scalability

• Availability of complete tools

• Shifting in the development process
MBT: In the V-Model

requirements  →  acceptance testing

specification  →  system testing

system design  →  integration testing

module design  →  model-based testing

coding  →  unit testing

modelling
MBT : Some Tools

- AETG
- Agatha
- Agedis
- Autolink
- Axini Test Manager
- Conformiq
- Cooper
- Cover
- DTM
- G∀st
- Gotcha
- JTorX
- MaTeLo
- MBTsuite
- MBTsuite
- NModel
- ParTeG
- Phact/The Kit
- QuickCheck
- Reactis
- Recover
- RT-Tester
- SaMsTaG
- Smartesting CertifityIt
- Spec Explorer
- Statemate
- STG
- Temppo
- TestGen (Stirling)
- TestGen (INT)
- TestComposer
- TestOptimal
- TGV
- TorX
- TorXakis
- T-Vec
- Uppaal-Cover
- Uppaal-Tron
- Tveda
- ............
MBT Tools: Off-Line - On-Line

- System model
- Model-based test generation
- Test execution
- SUT
- Pass fail

MBT Tools: Off-Line
- MBT Tools: On-Line
MBT Tools: Off-Line = Batch

test cases

model-based test generation

system model

test cases

test execution

SUT

pass fail
MBT Tools: On-Line = On-the-Fly

model-based test generation

system model

SUT

pass fail

On-Line = On-the-Fly
MBT Future: Current Research

• Partial and incomplete models
  – uncertainty in models and systems

• Test case selection
  – testing based on user profiles
  – risk-based MBT

• Integration of testing methods and tools
  – category partitioning, state-based, ioco, risk-based,
    combinatorial, constraint-based test-data generation,
    statistical test generation, . . . . .

• Model learning
  – automatic generation of models from tests
Complexity of Models → Learning
MBT

Questions?