Model-Based Testing

Theory

Tools

Applications

Jan Tretmans

ESI – Embedded Systems Innovation by TNO
Radboud University Nijmegen
Högskolan i Halmstad
jan.tretmans@tno.nl
Jan Tretmans

ESI

Embedded Systems Innovation by TNO
Eindhoven
The Netherlands

Radboud University
Nijmegen
The Netherlands
Embedded Systems Innovation
by TNO

Vision:
“Create economic and societal impact & value by embedded systems technology”

Mission:
“To advance industrial innovation and academic excellence in embedded systems engineering”
ESI WITHIN TNO

TNO THEMES AND INNOVATION AREAS

OUR THEMES

HEALTHY LIVING
INDUSTRIAL INNOVATION
DEFENCE, SAFETY & SECURITY
ENERGY
TRANSPORT & MOBILITY
BUILT ENVIRONMENT
INFORMATION SOCIETY

OUR INNOVATION AREAS

HEALTHY FOR LIFE
FOOD AND NUTRITION
WORK AND EMPLOYMENT
BIONEEDICAL INNOVATIONS

HIGH TECH SYSTEMS AND MATERIALS
SUSTAINABLE CHEMICAL INDUSTRY
SPACE

DEFENCE RESEARCH
SAFETY AND SECURITY RESEARCH

OIL AND GAS
ENERGY EFFICIENCY
GEOLOGICAL SURVEY OF THE NETHERLANDS
MARITIME AND OFFSHORE

RELIABLE MOBILITY SYSTEMS
SAFE AND CLEAN TRANSPORT

URBAN DEVELOPMENT
BUILDING AND INFRASTRUCTURE

INFOSTRUCTURES
INFRASTRUCTURES
Embedding intelligence into physical products

Typical characteristics:

- Multi-disciplinary design
- Software complexity
- Physical environments
- Distributed or networked
- Constrained resources
- Critical applications
- Quality, reliability, testing
- System evolution
Research cooperation with leading Dutch high-tech multinational industries & SME’s

Research cooperation with all Dutch universities with embedded systems research

Research cooperation in EU projects
Model-Based Testing
Motivation
What do Dykes have to do with Quality of Embedded Systems?
What do Dykes have to do with Quality of Embedded Systems?
Embedded Systems
Quality of Embedded Systems

Software is brain of system
• software controls, connects, monitors almost any aspect of ES system behaviour
• majority of innovation is in software

**Software determines quality and reliability of Embedded System**
• often > 50% of system defects are software bugs
Software is Different

Software is different from hardware:

- non-continuous
- any bug is a design error
- adopting redundancy is useless
- no wear and tear
- no MTBF; what is software reliability?
Trends & Challenges

- complexity
- size
- connectivity
- systems-of-systems
- change
- variability
- evolvability
- uncertainty
- multi-disciplinarity
- heterogeneous components

quality challenges
Model-Based Testing
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.
Model-Based Testing

MBT

next step in test automation:

+ test generation
+ result analysis
1 : Manual Testing

1. Manual testing
2 : Scripted Testing

1. Manual testing
2. Scripted testing

- Manual testing
- Scripted testing
- Test cases
- Test execution
- SUT
- Pass
- Fail
3: Keyword-Driven Testing

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

1. Manual testing
2. Scripted testing
3. Keyword-driven testing
4. Model-based testing
MBT: next step in test automation

- Automatic test generation
  + test execution + result analysis
- More, longer, and diversified test cases
  more variation in test flow and in test data
- Model is precise and consistent test basis
  unambiguous analysis of test results
- Test maintenance by maintaining models
  improved regression testing
- Expressing test coverage
  model coverage
  customer profile coverage

detecting more bugs
faster and cheaper
MBT: Benefits?

MBT: State of the Art
- promising, emerging
- a number of successful applications
- many companies are experimenting

MBT: State of Practice
- lagging behind

Reasons
- technical
- tools
- organizational
- maturity of testing
- educational
- ...

MBT: State for the Future
(for High-tech Embedded Systems)
- ?

But ....
If doing MBT is so smart, why ain’t you rich?
Model-Based
Verification, Validation, Testing, . . . . .
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, Testing

informal requirements → validation → model → verification → model-based testing → real world

informal world

formal world

SUT
Verification and Testing

Model-based verification:
- formal manipulation
- prove properties
- performed on model

Model-based testing:
- experimentation
- show error
- concrete system

Verification is only as good as the validity of the model on which it is based.

Testing can only show the presence of errors, not their absence.
Code Generation from a Model

A model is more (less) than code generation:

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

\[ y \times y = x \]

model of \( \sqrt{x} \)

- specification of properties
  - rather than construction
- under-specification
- non-determinism
Model-Based Testing

The \textit{ioco} Theory

for Labelled Transition Systems
Model-Based Testing Tools
# MBT Tools

- AETG
- Agatha
- Agedis
- Autolink
- Axini Test Manager
- Conformiq
- Cooper
- Cover
- DTM
- fMBT
- G∀st
- Gotcha
- Graphwalker
- JTorX
- MaTeLo
- MBTsuite
- M-Frame
- MISTA
- NModel
- OSMO
- ParTeG
- Phact/The Kit
- PyModel
- QuickCheck
- Reactis
- Recover
- RT-Tester
- SaMsTaG
- Smartesting CertifyIt
- Spec Explorer
- StateMate
- STG
- tedeso
- Temppo
- TestGen (Stirling)
- TestGen (INT)
- TestComposer
- TestOptimal
- TGV
- Tigris
- TorX
- TorXakis
- T-Vec
- Tveda
- Uppaal-Cover
- Uppaal-Tron
- ............
## MBT Tools

<table>
<thead>
<tr>
<th>Category</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Manager</td>
<td>AETG, Agatha, Agedis, Autolink, Axini Test Manager, Conformiq, Cooper, Cover, DTM, fMBT, G∀st, Gotcha, Graphwalker, JTorX, MaTeLo, MBTsuite, M-Frame, MISTA, NModel, OSMO, ParTeG, Phact/The Kit, PyModel, QuickCheck, Reactis, Recover, RT-Tester, SaMsTaG, Smartesting CertifyIt, Spec Explorer, StateMate, STG, tedeso, Temppo, TestGen (Stirling), TestGen (INT), TestComposer, TestOptimal, TGV, Tigris, TorX, TorXakis, T-Vec, Tveda, Uppaal-Cover, Uppaal-Tron</td>
</tr>
</tbody>
</table>
Yet Another MBT Tool: TorXakis

- AETG
- Agatha
- Agedis
- Autolink
- Axini Test Manager
- Conformiq
- Cooper
- Cover
- DTM
- fMBT
- $G \forall st$
- Gotcha
- Graphwalker
- JTorX
- MaTeLo
- MBTsuite
- M-Frame
- MISTA
- NModel
- OSMO
- ParTeG
- Phact/The Kit
- PyModel
- QuickCheck
- Reactis
- Recover
- RT-Tester
- SaMsTaG
- Smartesting CertifyIt
- Spec Explorer
- StateMate
- STG
- tedeso
- Temppo
- TestGen (Stirling)
- TestGen (INT)
- TestComposer
- TestOptimal
- TGV
- Tigris
- TorX
- TorXakis
- T-Vec
- Uppaal-Cover
- Uppaal-Tron
- . . . . . . . . . . .
Trends & Challenges

- Complexity
- Size
- Connectivity
- Systems-of-systems
- Multi-disciplinarity
- Change
- Variability
- Evolvability
- Uncertainty
- Heterogeneous components

Trends & challenges
Model Based Testing

- abstraction
- concurrency
- parallelism
- state + complex data
- model composition
- test selection
- multiple paradigms integration
- uncertainty nondeterminism
- statistical usage profiles
- under specification
- partial specification
- heterogeneous components
- multi disciplinarity
- change variability
- evolvability
- connectivity
- complexity
- variability
- MBT : Next Step Challenges
- Next Step Challenges
- MBT
Model-Based Testing

TorXakis
TorXakis: LTS & ioco

- SUT ioco model
- sound → exhaustive
- SUT passes tests

LTS test generation

set of LTS tests

LTS model

input/output conformance ioco

SUT behaving as input-enabled LTS

pass fail
STS: Symbolic Transition Systems

STS: model
with data

[\[ n \geq 35 \] \rightarrow button1]

[\[ n \geq 50 \] \rightarrow button2]

money \ ? n :: int

tea

return \ ! n \ - \ 35

coffee

return \ ! n \ - \ 50

with data

Model

return

Coffee

Tea

Money

Buttons

Symbolic Transition Systems (STS)
STS: Symbolic Transition Systems

in ? n :: int
[[ n ≠ 0 ]]

out ! m :: int
[[ 0 < m < -n ]]

out ! m :: int
[[ 0 < m < n ]]
**sioco : Symbolic ioco**

Specification: IOSTS $S(\iota_S) = \langle L_S, l_S, \mathcal{V}_S, \mathcal{I}, \Lambda, \rightarrow_S \rangle$

Implementation: IOSTS $P(\iota_P) = \langle L_P, l_P, \mathcal{V}_P, \mathcal{I}, \Lambda, \rightarrow_P \rangle$

both initialised, implementation input-enabled, $\mathcal{V}_S \cap \mathcal{V}_P = \emptyset$

$\mathcal{F}_S$: a set of symbolic extended traces satisfying $[\mathcal{F}_S]_{\iota_S} \subseteq \text{Straces}((l_0, \iota))$;

$P(\iota_P) \text{sioco} \mathcal{F}_S S(\iota_S)$ iff

$\forall (\sigma, \chi) \in \mathcal{F}_S \ \forall \lambda_\delta \in \Lambda_U \cup \{\delta\} : \iota_P \cup \iota_S \models \exists_{\mathcal{I}_U} (\Phi(l_P, \lambda_\delta, \sigma) \land \chi \rightarrow \Phi(l_S, \lambda_\delta, \sigma))$

where $\Phi(\xi, \lambda_\delta, \sigma) = \bigvee \{ \varphi \land \psi \mid (\lambda_\delta, \varphi, \psi) \in \text{out}_s((\xi, \top, \text{id})_0 \text{after}_s(\sigma, \top)) \}$

Theorem 1.

$P(\iota_P) \text{sioco} \mathcal{F}_S S(\iota_S)$ iff $[P]_{\iota_P} \text{ioco} [\mathcal{F}_S]_{\iota_S} [S]_{\iota_S}$
MBT Tools

model-based test generation

system model

SUT

test execution

pass fail

TTCN

TTCN Test cases

pass fail

MBT Tools


MBT: Ingredients

- **off-line MBT**
- **test cases**
- **model-based test generation**
- **test execution**
- **system model**
- **SUT**
- **test harness**
- **verdict**
- **test result analysis**
- **pass fail**

**Ingredients**:
- Ideas
- Requirements

**Feedback Loop**:
- Ideas flow to requirements
- Requirements drive model
- Model leads to test generation
- Test generation produces test cases
- Test cases are executed
- Execution results in verdict
- Verdict influences test result analysis
MBT: On-the-Fly

- Ideas
- Requirements
- Test harness
- SUT
- Model based
test generation
+ execution
- Model
- System
- On-the-fly MBT
- Pass fail
- Verdict
- Test result
- Analysis
TorXakis: An On-the-Fly MBT Tool
Tic-Tac-Toe

Tic-Tac-Toe Rules

CHANDEF MyChannels

ENDDEF

MODELDEF TicTacToe

CHAN IN In

CHAN OUT Out

BEHAVIOUR

In 'X' => Out 'O'

ENDDEF

CNECTDEF Sut

CHAN OUT In

HOST "localhost" PORT 7890

ENCODE In 'X' -> ! 'X'

CHAN IN Out

HOST "localhost" PORT 7890

DECODE Out <= ? s

ENDDEF

model.txs
TorXakis: Overview

Models
- process-algebraic modelling language
- state-based control flow and complex data
- support for parallel, concurrent systems
- composing complex models from simple models
- non-determinism, uncertainty
- abstraction, under-specification

Tool
- on-line MBT tool

Applications
- several high-tech systems companies
- experimental level

But....
- research prototype
- poor usability

Current Research
- test selection
- variability, features
- modelling
- integration in process

Under the hood
- SMT solvers for constraints and data generation (via SMT-LIB: Z3, CVC4)
- testing theory: sioco on STS
- algebraic data-type definitions with rewriting
- Haskell
- LPE: Linear process equations
- Other MBT tools for testing (QuickCheck)
Model-Based Testing

Applications
Electronic Passport

New Passport

- Machine Readable Passport (MRP, E-passport)
- with chip (JavaCard), contact-less
- storage of picture, fingerprints, iris scan, .......
- access to this data protected by encryption and a new protocol
- few years ago released in EU

Our job: testing of e-passports

- emphasis on access protocol
  == exchange of request-respons messages between passport and reader (terminal)
MBT for E-Passports : Model
model-based test tool

TorXakis

test runs

java drivers adapter

pass fail

state-based model

e-passport & wireless reader

english specifications
MBT for E-Passports: Results

- Tested:
  - Basic Access Control (BAC)
  - Extended Access Control (EAC)
  - Active Authentication (AA)
  - Data Reading

- Tests up to about 2,000,000 test events
  - complemented with manual tests

- No error found …
MBT in High-Tech Embedded Systems
MBT in High-Tech Embedded Systems

**Systems**
- large, complex, system-of-systems
- complex state + complex data
- variability, product line
- not always up-to-date specifications
- compositional
- parallelism, under-specification
- uncertainty, non-determinism,

**SUT**
- testing on simulated SUT:
  - virtual system, digital twin

**Models**
- how to make models?
- who makes models?: Testers
- DSL (Domain Specific Languages)
- construct model from tests

**Testing**
- state of practice:
  - keyword-driven test automation
- instrumentation: existing
  - keyword-driven test automation
- test selection via usage-profiles
Model-Based Testing

Using TorXakis
Model-Based Testing

*Theory, Tools, Applications*

- **MBT:** the next step
  in test automation!?

- **The future of testing**
  is model-based!?

- **If not, what is**
  the alternative?