MODEL BASED TESTING, THEORY AND PRACTICE

TIBOR CSÖNDES, PHD
SENIOR SPECIALIST, SECTION MANAGER
TEST COMPETENCE CENTER
ERICSSON HUNGARY
OUTLINE

› What is Model Based Testing (MBT)
› MBT Tools Overview
› Titan, TTCN-3 Test Framework
› Case Studies
  – TCP case study with Qtronic and TITAN
  – TitanSim as Test Harness
› Experiences and Recommendations
TEST AUTOMATION

› Test automation: automation of test execution
› MBT: automation of test design (automatic test generation using a model)

MBT is a higher level problem solution

Trend

Happy advanced tester

Typical tester

Model Based Testing, Theory and Practice | Public | © Ericsson AB 2010 | 2010-05-03 | Page 3
Testing before the MBT

TCP Connection establishment

“The active open is performed by the client sending a SYN to the server. It sets the segment's sequence number to a random value A.

In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another random number, B.

Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value, and the acknowledgement number is set to one more than the received sequence number i.e. B + 1. “

```
module MyExample {
  type port PCOType_PT message {
    inout TCPMessages;
  }
  type component MTCType_CT {
    port PCOType_PT My_PCO;
  }
  testcase tc_TP#1() runs on MTCType_CT system MTCType_CT {
    map(mtc:My_PCO, system:My_PCO);
    My_PCO.send(t_SYN(A));
    alt { ..
      My_PCO.receive(t_SYN_ACK(A+1,B))
      My_PCO.send(t_ACK(A+1,B+1))
      setverdict(pass);
    }
    control {
      execute ( tc_HelloW() );
    }
  }
} 
```
Testing as it Used to Be (2/2)

Conventional Testing vs. Model Based Testing

module MyExample {
    type port PCOType_PT message {
        inout TCPMessages;
    }
    type component MTCType_CT {
        port PCOType_PT My_PCO;
    }

testcase tc_TP#1 ()
    runs on MTCType_CT system MTCType_CT
    {
        map(mtc:My_PCO, system:My_PCO);
        My_PCO.send(t_SYN(A));
        alt {
            [] My_PCO.receive(t_SYN_ACK(A+1,B))
            {
                My_PCO.send(t_ACK(A+1,B+1))
                setverdict(pass);
            }
        }
    }

testcase tc_HelloW () ;
}

TC#2

TC#3

...etc
TCP Connection establishment
“The active open is performed by the client sending a SYN to the server. It sets the segment’s sequence number to a random value A.

In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another random number, B.

Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value, and the acknowledgement number is set to one more than the received sequence number i.e. B + 1.”

TCP Connection establishment
“The active open is performed by the client sending a SYN to the server. It sets the segment’s sequence number to a random value A.

In response, the server replies with a SYN-ACK. The acknowledgment number is set to one more than the received sequence number (A + 1), and the sequence number that the server chooses for the packet is another random number, B.

Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value, and the acknowledgement number is set to one more than the received sequence number i.e. B + 1.”
WHAT IS MODEL BASED TESTING (2/2)

› Conventional Testing vs. Model Based Testing

Algorithmically generated Model

TestSuites

TC#N

TC#2

TC#1

Algorithmically generated

TC#1

Test Harness

TestHarness
THE MODEL

› Model description

Graphical based: FSM; Petri Net; Label Transition System

Mixed: Graphical Solution extended by a language (C++; java; C#...etc.)

Pure language based: C++; Java; C#: TTCN-3
THE ALGORITHM

› Magic Algorithm: describes how to generate the Test cases

Coverage:
- Traverse every state
- Traverse some state
- Traverse every link
- …etc
MODEL BASED TESTING ON FIELD

Pros and Cons of Model Based Testing
- Reduces fault slip through Design phase
- Testing phase

Model development of the Design and model development of the Testing could take place parallel
- model development for testing verifies the model of the design
- some faults could be found in the “development phase”
- Reduces development time
- Model Driven Engineering
MODEL BASED TESTING WHY?

› Pros and Cons of Model Based Testing
  – Reduces fault slip through
  – Maintenance

› If the specification changes

› Change all the affected TestCases + TestHarness
  › Complicated!!

› Modify the Model
  › Easier maintenance!!

TestSuites

Model

Test Harness
MBT TOOLS OVERVIEW

› A number of different tools exist
  › **online testing**, 
    - model-based testing tool connects “directly” to a system under test and tests it dynamically.
  › **offline generation of executable tests**
    - model-based testing tool generates test cases as a computer-readable asset that can be later deployed automatically.
  › **offline generation of manually deployable tests**
    - model-based testing tool generates test cases as a human-readable asset that can be later deployed manually
WHAT IS TTCN-3?

TTCN-3

- ETSI standardized language
- Mainly used for protocol testing (SIP, IP TCP, HTTP)
- High level abstract language
- Protocol independent way of working

```plaintext
module MyExample {
  type port PCOType_PT message {
    inout charstring;
  }
  type component MTCTYPE_CT {
    port PCOType_PT My_PCO;
  }
  testcase tc_HelloW () runs on MTCTYPE_CT system MTCTYPE_CT {
    map(mtc:My_PCO, system:My_PCO);
    My_PCO.send ("Hello, world!");
    setverdict (pass);
  }
  control {
    execute (tc_HelloW());
  }
}
```
WHAT IS TITAN?

› TTCN-3 Development Environment
› Compiles and executes the code written in TTCN-3 (+ ASN 1, C, C++)
› Runs on LINUX/Solaris based hosts
› Available as
  – Eclipse plugin
  – Stand alone graphical Tool
  – Stand Alone cmd line tool
› Many other useful features are available as stand alone products to support various protocols
CASE STUDIES: INTRODUCTION

› Goals
  – A number of different tools exist: evaluation is needed
  – Integration with TITAN

› System Under Test
  – TCP connection establishment

› Contents
  – Test Arrangement
  – Workflow
  – Tools
    › Conformiq Qtronic
    › Elvior MOTES
    › Microsoft Spec Explorer
  – Comparison
  – TitanSim as Test Harness
CASE STUDIES: TEST ARRANGEMENT

 › Generated tests: Abstract tests

 › Test harness
  - Transforms the abstract tests to concrete, executable tests
  - Needs to be developed by using:
    › Titan and protocol specific useful features

---

Model Based Testing, Theory and Practice  |  Public   |  © Ericsson AB 2010  |  2010-05-03  |  Page 16
CASE STUDIES:

› Tool
  - Eclipse plugin
  - Computational Server

› Input
  - Qtronic Modeling Language
    › UML state model
    › Java like
  - Test purposes
    › Requirement coverage
    › State coverage
    › Transition coverage
    › Conditional branch coverage

› Output
  - Message Sequence Charts (MSC)
  - Can be exported to
    › Built-in adaptors
      - TCL, HTML, TTCN-3
    › User can write his own adaptors

http://www.conformiq.com/
CASE STUDIES:

› Tool
  - Eclipse plugin
  - Still in BETA

› Input
  - UML
    › SUT state model
  - TTCN-3
    › SUT interface
    › Test data
    › Test configuration
  - Test purposes
    › All transitions
    › N-sequences of transitions
    › Selected elements

› Output
  - Test Sequences in TTCN-3
  - Reactive Tester in TTCN-3

http://www.elvior.ee/
CASE STUDIES:

› Tool
  - Integrated in Visual Studio 2010

› Input
  - Model assembly
    › Spec# (C# based)
    › Cord coordination scripts
  - Test Purposes
    › Random walk
    › Transition coverage
    › Shortest path

› Output
  - State graph: result of exploration
  - Can be exported in C# or XML
  - Adapter must be written for other languages in C#

CASE STUDIES:

› Primarily model-based testing tool for .NET developers
› Model program is not needed
› Adapter class contains functions of transitions
› Model is derived from Cord Scripts
› Spec Explorer → C# test cases → TTCN-3 test cases
› Test execution in TITAN
WHAT IS TITAN SIM?

› TitanSim is…
  – Function library written in TTCN-3
  – Like the set of packages in the java domain

› Goal: Provide centrally the most commonly used features to support the development of a Load test application

› Provides commonly used features, such as
  – Handling protocol specific data and logic
    › Data definitions, basic protocol functions, etc.
  – Generic data structures with algorithms
    › Hashmaps, Free-Busy Queue, etc
  – Definition of source code structure
GLUE CODE

- Glue code is manually developed domain specific TTCN-3 code orchestrating existing protocol components.
- Idea: Use as much as possible from TitanSim for the glue code → less glue code development time.
TitanSim as Test Harness

- Modeling
  - abstract models
  - Focus is on the features to be tested

- MBT Tool output
  - Abstract Test suite
  - Simplified test messages

- Test harness
  - Transformation from abstract test message to real ones
  - Common responsibilities
    - Transport protocol handling
    - Timeout/retransmission handling
    - Checksum calculation
    - Unique id generation
  - Can be complex → development is time consuming.
EXPERIENCES, RECOMMENDATIONS (1/2)

› Introduce MBT in Function Test first (not performance test, characteristics, measurements, load, stability, etc.)
› Select the “right” object
› MBT is a paradigm shift
› “Right” competences is required, training is needed
› New roles should be established within the test organisation, especially the model designer/”test architect”
› When designing the model, the tester shall not think in terms of test cases – the tester should, ultimately, only think of the system behaviour
› The tester must have a thorough understanding of the new functionality, and be involved in (and contribute to!) the development project already in the early stages
The model must be good if the output shall be good!
With inexperienced model designers, it is easy to introduce faults in the model
Save time and money! Estimated average ~20-30%
We get efficient re-use of models and libraries
We find faults in design that we don’t find with traditional testing
The generated test cases are “better” than the traditional test cases. The generated test cases cover several events (Model/Test requirements), while the traditional test cases normally only cover one event/situation
Start with a smaller, well defined, well encapsulated, area/functionality

Reviews of the model is important during the design!