Test System Modelling and TTCN-3 Code Generation

-Xinyi Chen, Kao Bi, Ji Wu
Software Engineering Institute,
Beihang University
Agenda

- Motivation
- Model-Driven Testing
- U2TP & Test System Model
- Test System Modelling Example
- TTCN-3 Code Generation
- Conclusion
Motivation

• How to capture the implicit idea in testers mind
• How to make test design easier to be understood and maintained
• How to improve efficiency of test system design and implementation
MDA (Model-Driven architecture)

- Model-driven architecture (MDA) is a software design approach for the development of software systems.
- It was launched by the Object Management Group (OMG) in 2001.
- It provides a set of guidelines for the structuring of specifications, which are expressed as models.
- Model-driven architecture is a kind of domain engineering, and supports model-driven engineering of software systems.
**MDT (Model-Driven Testing)**

- The philosophy of MDA can also be applied on test modelling, which is called Model-driven testing.
- In MDT, testers use test system model to analyze test requirement and design the test system.
- Test system model describes all aspects of the test data, test cases and the test configure.
- Finally, test system model (PIT & PST) can be transformed to test code.
Why MDT?

• Abstraction
  ▫ works on a high level of abstraction thus makes tester focus on test design
  ▫ easy to be understood and maintained

• Automation
  ▫ generate executable test code directly from test model
MDT Framework

Refine

Test Req.

Design

Test Design

Debug/Refine

Iterate

Meta-Model

Adapt & Deploy

Test Result

Execute

Test System
The UML 2.0 Testing Profile (U2TP) is a UML 2.0 profile for the testing.
has become an official OMG standard since March 2004.
defines a language for designing, visualizing, specifying, analyzing, constructing and documenting the artifacts of test systems.
U2TP Concepts

- U2TP introduces four logical concept groups covering the aspects:
  - Test Architecture
    - concepts for test structure and test configuration
  - Test Data
    - concepts for test data used in test procedures
  - Test Behavior
    - concepts for the dynamic aspects of test procedures and addressing observations and activities during a test
  - Time
    - concepts for a time quantified definition of test procedures
Test System Model

• How to define test system model
  ▫ U2TP is a test modelling language on a higher level
  ▫ TTCN-3 is focus on test execution details

• Meta-model
  ▫ Test Configure
  ▫ Test Data
  ▫ Test Behavior
Test System Meta-model: Test Configure
Test System Meta-model: Test Data
Test System Meta-model: Test Behavior
<table>
<thead>
<tr>
<th>Aspect</th>
<th>U2TP</th>
<th>Test System Meta-model</th>
<th>TTCN-3 Core Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Objective</td>
<td>TestObjective</td>
<td></td>
<td>testcase</td>
</tr>
<tr>
<td>Test Case</td>
<td>TestCase</td>
<td></td>
<td>function</td>
</tr>
<tr>
<td>Defaults</td>
<td>Default</td>
<td></td>
<td>function</td>
</tr>
<tr>
<td>Behavior</td>
<td>ExciteAction</td>
<td></td>
<td>send/call/reply</td>
</tr>
<tr>
<td></td>
<td>ResponseJudgeAction</td>
<td></td>
<td>receive/getcall/getreply</td>
</tr>
<tr>
<td></td>
<td>DecisionAction</td>
<td></td>
<td>if(...) {...}</td>
</tr>
<tr>
<td></td>
<td>LoopAction</td>
<td></td>
<td>while(...) {...}</td>
</tr>
<tr>
<td></td>
<td>RefAction</td>
<td></td>
<td>function call</td>
</tr>
<tr>
<td></td>
<td>ActionBlock</td>
<td></td>
<td>{...}</td>
</tr>
<tr>
<td>Verdict</td>
<td>ValidationAction</td>
<td></td>
<td>setverdict</td>
</tr>
<tr>
<td>Arbiter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheduler</td>
<td>TestScheduler</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Control</td>
<td>TestBehavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Zone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Zone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Configure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>TestSystem</td>
<td></td>
<td>module</td>
</tr>
<tr>
<td>SUT</td>
<td>SUT</td>
<td></td>
<td>(test system interface)</td>
</tr>
<tr>
<td>Test Context</td>
<td>TestArchitecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Configuration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Component</td>
<td>TestComponent</td>
<td></td>
<td>component</td>
</tr>
<tr>
<td>Interface</td>
<td>TestInterface</td>
<td></td>
<td>port</td>
</tr>
<tr>
<td>Data Pool</td>
<td>DataPool</td>
<td></td>
<td>group</td>
</tr>
<tr>
<td>Data Partition</td>
<td>DataPartition</td>
<td></td>
<td>group</td>
</tr>
<tr>
<td>Data Selector</td>
<td>DataSelector</td>
<td></td>
<td>function</td>
</tr>
<tr>
<td>Wildcards</td>
<td></td>
<td></td>
<td>? *</td>
</tr>
<tr>
<td>Coding Rules</td>
<td>CodingRule</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TTCN-3 Code Generation

1. Test System Meta-model
   - Mapping
   - Instance of
2. TTCN-3 Core Language
3. Test System Model
   - Transformation
4. TTCN-3 Code
Tools

- **Eclipse** plug-in based on **EMF** (Eclipse Modelling Framework) and **GMF** (Graphical Modelling Framework)
- Test System Modelling Tools
  - Test Data
  - Test Configure
  - Test Behavior
- **TTCN-3** Code Generation Tools
Example: Local DNS Server

Local Network Client

Host name

IP-address

Local Domain Name Server
Data Type
Configure
Behavior
Example: Non-Local DNS Query

Local Network Client → Local DNS

Hostname

IP address

Local DNS → Root NS

Ask for remote DNS

Get remote address

Root NS → Remote DNS

Remote DNS → Local DNS

IP address

Software Engineering Institute, Beihang University
Configure
Behavior

call test scenario rootDNSQuery
Behavior(2)

TestScenario: rootDNSQuery
TTCN-3 Code Generation

```tcc
testcase testcase1() runs on NS system
DNS {
    clientPort.send(queryA);
    t1.start(10.0);
    rootDNSQuery();
    alt {
        [ ] clientPort.receive(responseA) {
            t1.stop;
            setverdict(pass);
        }
        [ ] client.receive(others) {
            t1.stop;
            setverdict(fail);
        }
        [ ] t1.timeout {
            setverdict(inconc);
        }
    }
}
```
TTCN-3 Code Generation(2)

function rootDNSQuery() {
    rootDNS.rootDNSPort.receive(queryB);
    rootDNS.rootDNSPort.send(responseB);
    NS.nsPort.receive(queryC);
    NS.nsPort.send(responseC);
}
TTCN-3 Code Generation(3)

```tcl
# Code generation for DNS components

type port DNSPort message {
  inout DNSMessage
}

type component Client {
  port DNSPort clientPort
}

type component rootDNS {
  port DNSPort rootDNSPort
}

type component NS {
  port DNSPort nsPort
}

type component DNS{
  port DNSPort clientPort;
  port DNSPort rootPort;
  port DNSPort remotePort;
}

map(Client:clientPort, system:sutPort);
map(NS:nsPort, system:remotePort);
map(rootDNS:rootDNSPort, system:rootPort);
```
TTCN-3 Code Generation(4)

```plaintext
type enumerated MessageType
    {e_Question, e_Answer};

type record DNSMessage {
    integer identification,
    MessageType messageType,
    charstring question,
    charstring answer    optional
}
```

Software Engineering Institute, Beihang University
Conclusion

• propose a model driven testing approach
• test system model is defined to describe the test system in three aspects: test data, test configure and test behavior
• a set of tools are developed to visualize the test system model in the period of test design
• these tools can be used to model the static and dynamic characteristics of test system
• automatic generation of executable TTCN-3 from this test system model
Thank you

Contact:
Ji Wu: wuji@buaa.edu.cn
Xinyi Chen: era@sei.buaa.edu.cn