New TTCN-3 Test Adapter Framework
with Capability Description

Yang LIU
Bei Hang University
Outline

• Introduction
• The Problem
• The solution
• Test Adapter Framework
• Evaluation
• Conclusions
Introduction
The separation of test specification with test adapter

Test System User

TM: Management

SA (System Adapter)  |  TE  |  PA (Platform Adapter)

SA (System Adapter)  |  CD: Codec

CH: Component Handling

System Under Test (SUT)
Outline

• Introduction
• The Problem
• The solution
• Test Adapter Framework
• Evaluation
• Conclusions
The Problem

• The dependencies among the abstract test case, CD and adapters (SA and PA) are obvious.
• There is no explicit semantic definition of the dependencies in TTCN-3 standard so far.
Outline

- Introduction
- The Problem
- The solution
- Test Adapter Framework
- Evaluation
- Conclusions
The solution

• Capability description language for specifying the dependencies among the abstract test case, CD, SA and PA

• A Test Adapter Framework
  – Automatic load the Adapter which comply with the requirements of test cases
  – Based on the description language
  – Transparent to the test system
Outline

• Introduction
• The Problem
• The solution
• Test Adapter Framework
• Evaluation
• Conclusions
Mapping between the Adapter and Test Case

- Adapter Capacity
  - SA: Communication Adapter
  - PA: External method Adapter
  - CD (ECD): Data Adapter

- Test System Interface (TSI)'s demand
  - Demand for communications capacity
  - Demand for External method
  - Input and output data types
Adapter Framework – Overall Architecture

Test Adapter Framework

Function Layer
- Adapter 1
- Adapter 2
- ……
- Adapter N

Configuration Layer
- Configuration Management
- Switcher

Interactive Layer
- SA
- PA
- CD

Execute Engine

TE

SUT
Adapter Framework – Switcher

- SAs
- PAs
- CDs

Test Adapter Switcher

- Loader
- Mapper
- Selector
Adapter Framework - Loader

- Environment Description
  - OS
  - Environment Vars

- Test Adapter Description
  - SA
  - PA
  - CD
Adapter Framework - Mapper

SA
- port_type

TSI
- type : String

PA
- module

CD
- rule
  - encodeRules

TSI
- port_name
- type
- module
- Encode Rules
  - rule

SA
- name
- port_type

PA
- name
- module

CD
- name
- rule

SA (jar)
PA (jar)
CD (jar)
example of capability matching

capability requirement description of the test case:

```
- <testcase Name="tc001" Module="TestCases">
  - <tsiPort>
    <name>nameOfCA4thisTSI02</name>
  - <encdecRule>
    <encodeRules>
      <encodeRuleName2>encodeRuleName2</encodeRuleName2>
    </encodeRules>
  - <typeNameOfthiPort02></typeNameOfthiPort02>
  </tsiPort>
</testcase>
```

capability provision description of the CD & SA:

```
- <codec name="Codec_02">
  <jarfile>cdJarFile2.jar</jarfile>
  <class>Codec2.class</class>
- <encoderule name="encodeRuleName2">
  <encodemessagetype>MessageType2De</encodemessagetype>
  <decodemessagetype>MessageType2De</decodemessagetype>
</encoderule>
- <lib>
  <jarfile>LibFile4CD03.jar</jarfile>
  <jarfile>LibFile4CD04.jar</jarfile>
</lib>
</codec>

- <SA name="SA_02">
  <jarfile>saJarFile2.jar</jarfile>
  <class>saClass2.class</class>
  <porttype>
    <typeNameOfthiPort02></typeNameOfthiPort02>
  </porttype>
- <lib>
  <jarfile>dependentJarFile.jar</jarfile>
  <jarfile>dependentJarFile2.jar</jarfile>
</lib>
</SA>
```
Adapter Framework – Selector

Diagram:
- Selector
- TE
- Mappings
- Entity

1: getAttribute
2: attribute
3: queryEntity
4: entityAttribute
5: call
6: return
Dynamic switcher

- Prerequisite:
  - TTCN-3 test script
  - SA, PA & CD
  - Test adapter & TSI’s Description File
Outline

• Introduction
• The Problem
• The solution
• Test Adapter Framework
• Evaluation
• Conclusions
Evaluation

• Experiment:
  • SUT
    – Simulated heterogeneous distributed systems
  • Test system
    – SA and CD of Three test subsystems
    – test script
  • Support Tools
    – Compiler (TTthree)
    – Execution Engine (TTman)
## Experiment with TAF

<table>
<thead>
<tr>
<th>SUT</th>
<th>TCP</th>
<th>UDP</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>TCPTest</td>
<td>UDPTest</td>
<td>HTTPTest</td>
</tr>
<tr>
<td>TSIPortName</td>
<td>tcpSystemPort</td>
<td>udpSystemPort</td>
<td>httpPortArray</td>
</tr>
<tr>
<td>type</td>
<td>TCP</td>
<td>UDP</td>
<td>HTTP</td>
</tr>
<tr>
<td>encodeRules</td>
<td>TCPCodec</td>
<td>UDPCodec</td>
<td>HTTPCodec</td>
</tr>
<tr>
<td>SA</td>
<td>TCP_TA.jar /TCPTestAdapter</td>
<td>UDP_TA.jar /UDPTestAdapter</td>
<td>HTTP_TA.jar /HTTPTestAdapter</td>
</tr>
<tr>
<td>CD</td>
<td>TCP_Codec.jar /TCPCodec</td>
<td>UDP_Codec.jar /UDPCodec</td>
<td>HTTP_Codec.jar /HTTPCodec</td>
</tr>
</tbody>
</table>
TAF – Test Result

21:09:39.781: parsing given loader file 'main.elf'
21:09:39.062: starting test cases...

21:09:39.062: starting test case 'http_tc' -> 1

HTTPTestAdapter: Request <to: /127.0.0.1/localhost:8080/dinolist.xml

HTTPTestAdapter: Response

File size: 3192
Warning: validation was turned on but an org.xml.sax.ErrorHandler was not set, which is probably not what is desired. Parser will use a default ErrorHandler to print the first 10 errors. Please call the setErrorHandler method to fix this.
Error: URI = "null", Line = "42". : Document root element "dinolist", must match DOCTYPE root "null".
Error: URI = "null", Line = "42". : Document is invalid: no grammar found.

21:09:39.328: http_tc -> pass

21:09:39.343: starting test case 'tcp_tc' -> 2

MyTestAdapter: Sending <to: /127.0.0.1/103TCP

0103TCP

MyTestAdapter: Received (7 chars)

0103TCP

21:09:39.406: tcp_tc -> pass


UDPTestAdapter: Sending <to: /127.0.0.1/103TCP

XXX XXX 0x01 XXX 0x04
0x10 0x10 0x10 0x10

UDPTestAdapter: Received (8 bytes)

XXX XXX 0x01 XXX 0x04
0x10 0x10 0x10 0x10

UDPTestAdapter: Received (8 bytes)

Total number of executed test cases: 3
none: 0
pass: 3
incon: 0
fail: 0
error: 0
Performance comparison

- HTTP
- TCP
- UDP
- Total

Execution Time (seconds)

- Traditional independent test
- Dynamic switching adapter test
Outline

• Introduction
• The Problem
• The solution
• Test Adapter Framework
• Evaluation
• Conclusions
Conclusions

• Why we design the test adapter framework
• Existing problems
  – Test Adapter Framework configuration is complex
  – The efficiency of loading the modules which will be dynamic switching is low
Conclusions

• Future work
  – Raise the ability to match to the semantic level
  – Use a general language of Capability Description
  – Set the general test adapter Standard
THANK YOU