TTCN-3

(Testing and Test Control Notation)
Overview

• Motivation
• History
• Introduction (TTCN-3)
• Example + Basics (Core Language)
• Test System Architecture
• U2TP and TTCN-3
• Conclusion
Motivation

• One test notation for black-box and grey-box testing (SUT)
• To have a universally understood test language
• Standardized test specifications and implementation
• The majority of tests is still developed manually
• To develop test cases at an abstract level
• Specify detailed test descriptions on different levels of abstraction including (component, integration, system level testing)
• Reduces the costs for education and documentation
History

- TTCN (Tree and Tabular Combined Notation) 1992 as part 3 of the ISO/IEC standard 9646
- “OSI Conformance Testing Methodology and Framework”
- TTCN was designed for testing OSI protocols
- TTCN: (GSM, DECT, INAP, N-ISDN, B-ISDN)
- Limited usability (interoperability, robustness, regressions, and system testing)
- 1998 ETSI was asked for a new test language
- 2000 first version of TTCN-3
- Since then it is Continuously maintained test technology
- 2007 TTCN-3 v3.2.1 by ETSI

European Telecommunications Standards Institute (ETSI)
7 parts

Part 1: Core Language (textual syntax of TTCN-3)

Part 2: Tabular Presentation Format (TTCN-3 specification within a collection of tables)

Part 3: Graphical Presentation Format (represents interaction between the SUT and the test system. It is based on Message Sequence Chart (MSC))

Part 4: Operational semantics (describes the meaning of TTCN-3 behavior)

Part 5: TTCN-3 Runtime Interface (TRI) (API to adapt TTCN-3 test system to an SUT)

Part 6: TTCN-3 Control Interface (TCI) (handles the adaption of the test system)

Part 8: Use of IDL in TTCN-3
Different types of testing

• Functional testing
• Load testing
• Scalability testing
• Regression testing
• Robustness testing
• Interoperability testing
Example

Description:

- Request (URL) to a web server
- The web server responds with an XML file
- The XML file should contain a list of dinosaurs
- If the file contains a dinosaur with the name Brachiosaurus the test verdict is pass.
Example (GFT)

function ptcBehaviour ( )

Runs on ptc Type

self

httpPort

ptcType

httpPortType

urlTemplate

localTimer

DinoListTemplate

localTimer

pass

fail

localTimer

fail

localTimer
Concepts

• Look and feel like C or C++
• Black Box Testing (SUT)
• Data -> SUT -> Reactions
• Test specification and implementation
• Contains all important features for testing
  • Test verdicts
  • Matching mechanism
  • Time restrictions
  • Ability to specify encoding information
  • Support for different kinds of communication
  • Possibility to log test information
• Interfaces (ports)
• TTCN-3 test specification is defined by a set of modules
Module Definitions

**Imports**
Importing definitions from other modules defined in TTCN-3 or other languages

**Data Types**
User defined data types (messages, PDUs, information elements, ...)

**Test Data**
Test data transmitted/received during test execution (templates, values)

**Test Configuration**
Definition of the test components and communication ports

**Test Behavior**
Specification of the dynamic test behavior

**Module Control**
Defining the sequence, loops, conditions, etc. for the execution of test cases
TTCN-3 Data Types and Templates

Keyword: **type**
Syntax: **type** <basic/special type> name
Basic types: boolen, iteger, float, charstring, etc.
Special types: verdicttype (none, pass, inconc, fail), default, address

Structured types Syntax: **type** <structured type> name {}  
Structured types: record, set, enumerated

Keyword: **template**
Syntax: **template** <TemplateRestriction> name
? ... anytype
Example (Core Language)

```plaintext
module dinolistTest {
    modulepar integer NUMBER_OF_PTCs := 1
    with {
        extension (NUMBER_OF_PTCs)
            "Description: Default number of PTCs";
    }
    type record urlType {
        charstring protocol,
        charstring host,
        charstring file
    }
    template urlType urlTemplate := {
        protocol := "http://",
        host := "www.testingtech.de",
        file := "/TTCN-3_Example/dinolist.xml"
    }
};
```
Example (Core Language)

```plaintext
: 
type set of dinosaurType dinolistType;
type record dinosaurType {
  charstring name,
  charstring len,
  charstring mass,
  charstring time,
  charstring place
}
template dinosaurType BrachiosaurusTemplate := {
  name := "Brachiosaurus",
  len := ?,
  mass := ?,
  time := ?,
  place := ?
}
template dinolistType DinoListTemplate := {
  ?,
  ?,
  BrachiosaurusTemplate,
  ?,
  ?,
  ?
}
:
```
TTCN-3 Communication

- Message-based
  - Communication via buffers
  - Asynchronous communication
  - `send`: to send a message
  - `receive`: to receive a message
  - `trigger`: to discard all messages until the specified message is received

- Procedure-based communication
  - Remote procedure calls for communication
  - Synchronous communication
  - `call`: to invoke a remote procedure
  - `getcall`: to accept a call from remote
  - `reply`: to reply to a previously received call
  - `raise`: to report an exception to a previously received call
  - `catch`: to collect an exception reported by a remote procedure invocation
TTCN-3 Ports

Keyword: **port**
Example: type **port** httpPortType **message/procedure** {
    in MyMsgA/MyProcedureA;
    out MyMsgB/MyProcedureB;
}

- Dynamic binding
- One to many possible
- Connection between PTCs
- Connection between PTC and SUT
- Difference between mapped and connected ports
- Connected ports are used for the communication with other test components
- Mapped ports are used for the communication with the SUT
TTCN-3 Components (1)

Keyword: component
There are three kinds of components:

• One Main Test Component (MTC)
  • Is created and stared automatically
  • Behaviour is specified in the body of the test case def.
  • A test case terminates when the MTC terminates
    (includes PTC termination)

• Non / many Parallel Test Component (PTC)
  • Created, started and stopped dynamically

• System Component (SUT, no local Timer, constants or variables)
TTCN-3 Components (2)

Components Contain:
• Ports
• Timers
• Local variables
• Constants

Methods:
• Start
• Stop
• Done
• Running
• Create

Each test component maintains its own local test verdict!
Example (Core Language)

```
:  

type component ptcType {  
    port httpPortType httpPort;  
    timer localTimer := 3.0;  
}

}  

type port httpPortType message {  
    out urlType;  
    in dinolistType;  
}

type component mtcType {}  

type component systemType {  
    port httpPortType httpPortArray [NUMBER_OF_PTCS];  
}

:
```
TTCN-3 Test cases

Keyword: **testcase**
Syntax: **testcase** name()

```plaintext
runs on ComponentType  system ComponentType { ... }
```

A test case
• Defines the behavior of the MTC
• Can contain `alt` statements
• Ends with a verdict
Example (Core Language)

```plaintext
: testcase DinoListTest_1()
runs on mtcType system systemType {
  var ptcType ptcArray [NUMBER_OF_PTCs];
  var integer i := 0;
  for (i := 0; i < NUMBER_OF_PTCs; i := i + 1) {
    ptcArray[i] := ptcType.create;
    map (ptcArray[i]: httpPort, system: httpPortArray[i]);
  }
  for (i := 0; i < NUMBER_OF_PTCs; i := i + 1) {
    ptcArray[i].start (ptcBehaviour());
  }
  all component.done;
}
```
TTCN-3 Functions

Keyword: `function`
Syntax: `function` name()
    `runs on` ComponentType { ... }

• Can structure the test system behavior
• Contain `alt` statements
Example (Core Language)

```plaintext
function ptcBehaviour() runs on ptcType {
    httpPort.send (urlTemplate);
    localTimer.start;
    alt {
        [] httpPort.receive (DinoListTemplate) {
            localTimer.stop;
            setverdict (pass);
        }

        [] httpPort.receive {
            localTimer.stop;
            setverdict (fail);
        }

        [] localTimer.timeout {
            setverdict (fail);
        }
    }
}
```
Example (Core Language)

```plaintext
: control {
  execute (DinoListTest_1 ());
}
```
Example (GFT)

Testcase DinoListTest_1
runs on mtcType system systemType

mtc

var ptc Type ptcArray[NUMBER_OF_PTCS]

var integer i:= 0

for (i:=0; i<NUMBER_OF_PTCS; i:=i+1)

ptcArray[i] := ptcType.create

map (ptcArray[i]: httpPort, system:httpPortArray[i])

for (i:=0; i<NUMBER_OF_PTCS; i:=i+1)

ptcArray[i].start(ptcBehaviour)

All component .done
Example (GFT)

function ptcBehaviour()

Runs on ptc Type

self

ptcType

httpPort

httpPortType

urlTemplate

localTimer

alt

DinoListTemplate

localTimer

pass

fail

fail

localTimer
TTCN-3 Test System Architecture

TTCN-3 tests can be distributed over several test devices!

TCI ... TTCN-3 Control Interfaces
TRI ... TTCN-3 Runtime Interfaces
(TCI and TRI operations are defined in IDL)
TE ... TTCN-3 Executable
TM ... Test Management
CH ... Component Handling
CD ... Codec
SA ... System Adapter
PA ... Platform Adapter
## U2TP and TTCN-3

### U2TP ... UML 2.x testing profile

<table>
<thead>
<tr>
<th></th>
<th>U2TP</th>
<th>TTCN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Specification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Execution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test Meta Modelling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format</td>
<td>Graphical</td>
<td>Textual and graphical</td>
</tr>
<tr>
<td>Transformation</td>
<td>U2TP to TTCN-3</td>
<td>TTCN-3 to U2TP</td>
</tr>
</tbody>
</table>

### U2TP makes UML applicable for the design of test systems

### U2TP concepts:
- Test architecture
- Test behaviour
- Test data
- Time

GFT is the archetype for U2TP
Conclusion

- Independent Core Language
- Can handle external data
- TFT and GFT for in and output
- Used for black and grey box testing
- Platform independent
- One language to have a universally understood test language

Thank you for your attention!
References

Ina Schieferdecker, Jens Grabowski, Theofanis Vassiliou-Gioles and George Din – *The Test Technology TTCN-3*
Springer-Verlag Berlin Heidelberg 2008