1st UCAAT

ETSI’s User Conference on Advanced Automated Testing
1st User Conference on Advanced Automated Testing

Introduction to TTCN-3

Paris, October 2013
Theofanis Vassiliou-Gioles
Founder and CEO of Testing Technologies
vassiliou@testingtech.com
www.testingtech.com

Master in Electrical Engineering
Started communication testing 1996
ATM test specification Standardization
ETSI TTCN-3 Standardization
Application of test automation in new domains
Agenda

- Motivation
- Introduction to TTCN-3 – TTCN-3 by Example
The TTCN-3 Language

Introduction to TTCN-3

Motivation
How Much Does Testing Cost?

“... the national annual cost estimates of an inadequate infrastructure for software testing are estimated to be $59.5 billion. The potential cost reduction from feasible infrastructure improvements is $22.2 billion."

The Economic Impacts of Inadequate Infrastructure for Software Testing

Study by NIST, May 2002
And today?

World Quality Report 2013-14

As consumers demand high performance, error-free applications, organizations are increasing their QA budgets and more testing functions are centralized.

Focus on Testing is growing everywhere...
A higher share of the IT budget is invested in Testing ...

23% of the IT budget is spent on QA & Testing compared to 18% last year -

46% of QA budget is spent on "transformational projects" - compared to 41% last year -

... organizations are industrializing and outsourcing their QA...

41% of projects are done entirely in-house - compared to 51% last year -

26% of organizations have a centralized testing function - up from 8% in 2012 -

19% of businesses have fully operational Test Centers of Excellence - compared -

...and as mobile applications increase, mobile testing gains traction...
**PRIMARY FOCUS**

- #1 Efficiency and Performance: 59%
- #2 Security: 56% (up from 18% last year)

**BIGGEST CHALLENGE**

Lack of appropriate processes/methods

- 34%
- 56%

...and cloud-based testing is expected to increase.

By 2015, 32% of Testing will be performed in the Cloud

30% of cloud-based testing is performed on critical, externally facing applications (up from 20% last year)
Spendings in Testing (WQR 2013)

- Hardware: 40%
- Software: 28%
- Rest / People: 32%
Testing Today

- Is
  - Important
  - Expensive
  - Time critical

- But
  - Only rarely practiced as a strategic component
  - Unsystematic throughout the organization
  - Performed by hand
  - Error-prone
  - Uncool („If you are a bad programmer you might be a tester.“)
  - Unconstructive
Why Using TTCN-3 (2)

- Speed to Market
- High Integration
- IP
- High Reliability
- 3G/4G
- WLAN
- High Quality
- MATCH
- Law Changes
- Complexity
- Optimal Cost
- Requirements
- Test Methods
History (1)

- **TTCN (1992)**
  - Published as an ISO standard
  - Tree and Tabular Combined Notation
  - Used for protocol testing only
    - GSM, N-ISDN, B-ISDN

- **TTCN-2/2++ (1997)**
  - Concurrent tests
  - Modularization
  - Manipulate external data
  - Rather for conformance testing
  - Developed by ETSI MTS
History (2)

TTCN-3 (2000)

- Testing and Test Control Notation
- Developed by ETSI MTS
- Standard language
  - Well defined syntax and semantics
- Enhanced communication, configuration and control
- Standard test specification
  - SIP, SCTP, M3UA, IPv6
  - HiperLan, HiperAccess, Wimax
  - 3GPP LTE, OMA
  - TETRA
  - MOST, AUTOSAR
Since 2002 standard bodies using TTCN-3 to define test specifications

- ETSI 3GPP
- WiMAX Forum
- OMA
- TETRA
- AUTOSAR
- MOST

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>Start of developing TTCN-3</td>
</tr>
<tr>
<td>2000</td>
<td>TTCN-3 Standard /TT Foundation</td>
</tr>
<tr>
<td>2002</td>
<td>ETSI: SIP</td>
</tr>
<tr>
<td></td>
<td>ETSI: HiperACCESS</td>
</tr>
<tr>
<td>2003</td>
<td>ETSI: HiperLAN2</td>
</tr>
<tr>
<td>2004</td>
<td>ETSI: HiperMAN</td>
</tr>
<tr>
<td>2005</td>
<td>WiMAX Forum: 802.1d</td>
</tr>
<tr>
<td></td>
<td>ETSI IPv6</td>
</tr>
<tr>
<td>2006</td>
<td>WiMAX Forum: 802.1e</td>
</tr>
<tr>
<td></td>
<td>AUTOSAR: Pilot</td>
</tr>
<tr>
<td></td>
<td>WiMAX Forum: Interop</td>
</tr>
<tr>
<td>2007</td>
<td>ETSI: VoIP/POTS Testing</td>
</tr>
<tr>
<td></td>
<td>OMA: Interop Pilot</td>
</tr>
<tr>
<td>2008</td>
<td>AUTOSAR: Rollout</td>
</tr>
<tr>
<td>2011</td>
<td>OMA: Rollout</td>
</tr>
<tr>
<td>2012</td>
<td>TETRA: Pilot / Rollout</td>
</tr>
<tr>
<td></td>
<td>EUROCONTROL: Rollout</td>
</tr>
</tbody>
</table>
Maintenance of TTCN-3

- Standard is constantly maintained
  - Through Change Requests (CRs)
  - Extension proposals
  - Active contributions in the TTCN-3 community
    - TTCN-3 mailing list, TTCN-3 users conference

- ETSI STFs (Specialist Task Force)

- Change requests result in new editions of the standard
  - 2000: Edition 1
  - 2003: Edition 2
  - 2005: Edition 3
  - 2010: Edition 4.2.1
  - 2011: Edition 4.3.1
  - 2012: Edition 4.4.1
  - 2013: Edition 4.5.1

Testing is

- a technical process
- performed by experimenting with a software product
- in a controlled environment
- following a specified procedure
- with the intent of observing one or more characteristics of the product
- by demonstrating the deviation of the product’s actual status from the required status/specification
Testing Today’s Systems

- Component-based
  - Test-components contribute to SUT functionality and performance

- Distributed
  - Not only local, but also distributed test setups

- Dynamic in terms of behavior and configuration
  - Testing of static and dynamic aspects; dynamic creation of test components

- Use various type systems to exchange data
  - Open to all type systems

- Service is essential
  - Concentration on service-oriented black-box testing
Design Principles of TTCN-3

- One test technology for different kind of testing
  - Distributed, platform-independent testing
  - Integrated graphical test development, documentation and analysis
  - Adaptable, open test environment

- One test technology for distributed IT and telco systems and beyond
Main Aspects of TTCN-3

Triple C
- Configuration: Dynamic concurrent test configurations with test components
- Communication: Various communication mechanisms (message-based, procedure-based)
- Control: Test case execution and selection mechanisms

Features
- Well-defined syntax, static and operational semantics
- Different presentation formats
- Module concept
- Extendibility via attributes, external function, external data
- Harmonization with ASN.1, integration of XML, IDL, …
TTCN-3 Standards

- ETSI ES 201 873-1 TTCN-3 Core Language (CL)
- ETSI ES 201 873-2 TTCN-3 Tabular Presentation Format (TFT)
- ETSI ES 201 873-3 TTCN-3 Graphical Presentation Format (GFT)
- ETSI ES 201 873-4 TTCN-3 Operational Semantics
- ETSI ES 201 873-5 TTCN-3 Runtime Interface (TRI)
- ETSI ES 201 873-6 TTCN-3 Control Interfaces (TCI)
- ETSI ES 201 873-7 Integration of ASN.1
- ETSI ES 201 873-8 Integration of IDL
- ETSI ES 201 873-9 Integration of XML
- ETSI ES 201 873-10 T3Doc
- ETSI ES 202 781 TTCN-3 Extension: Configuration And Deployment Supp
- ETSI ES 202 782 TTCN-3 Extension: Performance & Real-Time Testing
- ETSI ES 202 784 TTCN-3 Extension: Advanced Parametrization
- ETSI ES 202 785 TTCN-3 Extension: Behaviour Types
- ETSI ES 202 786 TTCN-3 Extension: Continuous Signals
- ETSI ES 202 789 TTCN-3 Extension: Extended TRI

- Maintenance on the basis of change requests by ETSI
- Standard available for download at http://www.etsi.org
- Testing Tech tools support Edition 4.5.1
- Also standardized by the ITU-T as ITU-T Z.140 series
TTCN-3 Standards

- ETSI ES 201 873-1  TTCN-3 Core Language (CL)
- ETSI ES 201 873-2  TTCN-3 Tabular Presentation Format (TFT)
- ETSI ES 201 873-3  TTCN-3 Graphical Presentation Format (GFT)
- ETSI ES 201 873-4  TTCN-3 Operational Semantics
- ETSI ES 201 873-5  TTCN-3 Runtime Interface (TRI)
- ETSI ES 201 873-6  TTCN-3 Control Interfaces (TCI)
- ETSI ES 201 873-7  Integration of ASN.1
- ETSI ES 201 873-8  Integration of IDL
- ETSI ES 201 873-9  Integration of XML
- ETSI ES 201 873-10 T3Doc
- ETSI ES 202 781  TTCN-3 Extension: Configuration And Deployment Supp
- ETSI ES 202 782  TTCN-3 Extension: Performance & Real-Time Testing
- ETSI ES 202 784  TTCN-3 Extension: Advanced Parametrization
- ETSI ES 202 785  TTCN-3 Extension: Behaviour Types
- ETSI ES 202 786  TTCN-3 Extension: Continuous Signals
- ETSI ES 202 789  TTCN-3 Extension: Extended TRI

Maintenance on the basis of change requests by ETSI
Standard available for download at http://www.etsi.org
Testing Tech tools support Edition 4.5.1
Also standardized by the ITU-T as ITU-T Z.140 series
TTCN-3 Standards

- ETSI ES 201 873-1: TTCN-3 Core Language (CL)
- ETSI ES 201 873-2: TTCN-3 Tabular Presentation Format (TFT)
- ETSI ES 201 873-3: TTCN-3 Graphical Presentation Format (GFT)
- ETSI ES 201 873-4: TTCN-3 Operational Semantics
- ETSI ES 201 873-5: TTCN-3 Runtime Interface (TRI)
- ETSI ES 201 873-6: TTCN-3 Control Interfaces (TCI)
- ETSI ES 201 873-7: Integration of ASN.1
- ETSI ES 201 873-8: Integration of IDL
- ETSI ES 201 873-9: Integration of XML
- ETSI ES 201 873-10: T3Doc
- ETSI ES 202 781: TTCN-3 Extension: Configuration And Deployment Supp
- ETSI ES 202 784: TTCN-3 Extension: Advanced Parametrization
- ETSI ES 202 785: TTCN-3 Extension: Behaviour Types
- ETSI ES 202 786: TTCN-3 Extension: Continuous Signals
- ETSI ES 202 789: TTCN-3 Extension: Extended TRI

Maintenance on the basis of change requests by ETSI
Standard available for download at http://www.etsi.org
Testing Tech tools support Edition 4.5.1
Also standardized by the ITU-T as ITU-T Z.140 series
The TTCN-3 Language

Introduction to TTCN-3

TTCN-3 by Example
TTCN-3 By Example

Tester

Local Network
Client

Main Test Component

Send fully qualified hostname

Return IP-address

Local Domain
Name Server

System Under Test
TTCN-3 By Example

Tester

Client

(main test component)

pass

(www.testingtech.com,A)

(DNS)

(www.testingtech.com,212.227.57.158,A)

System Under Test
## Generic Protocol Architecture(s)

<table>
<thead>
<tr>
<th>OSI View</th>
<th>TCIP/IP View</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 / Phy</td>
<td>TCP, UDP, SCTP</td>
</tr>
<tr>
<td>L2 / Data Link</td>
<td>IPv4, IPv6</td>
</tr>
<tr>
<td>L3 / Network</td>
<td>Ethernet, Token Bus, Token Ring, FDDI, IPoAC</td>
</tr>
<tr>
<td>L4 / Transport</td>
<td>HTTP, FTP, SMTP, POP, Telnet, DNS</td>
</tr>
<tr>
<td>L5 / Session</td>
<td>TCP, UDP, SCTP</td>
</tr>
<tr>
<td>L6 / Presentation</td>
<td>IPv4, IPv6</td>
</tr>
<tr>
<td>L7 / Application</td>
<td>Ethernet, Token Bus, Token Ring, FDDI, IPoAC</td>
</tr>
</tbody>
</table>

**OSI View**
- **Application Layer**
- **Transport Layer**
- **Internet Layer**
- **Link Layer**

**TCIP/IP View**
- **TCP, UDP, SCTP**
- **IPv4, IPv6**
- **Ethernet, Token Bus, Token Ring, FDDI, IPoAC**
Generic Protocol Architecture(s)

OSI View

L7/ Application
L6 / Presentation
L5 / Session
L2 / Transport
L3 / Network
L2 / Data Link
L1 / Phy

TCIP/IP View

Application Layer
Transport Layer
Internet Layer
Link Layer
Data

Application

OSI View

L7/ Application
L6 / Presentation
L5 / Session
L2 / Transport
L3 / Network
L2 / Data Link
L1 / Phy

TCIP/IP View

Application Layer
Transport Layer
Internet Layer
Link Layer
Data

Application
When we test we ...

- Select the **protocol** or **application** to test
  → DNS

- Select the **test access**
  → UDP, IPv4, Ethernet
When we test we would like to ...

- **Concentrate** on the protocol (application) on an **abstract** level
- **Do not care** for the concrete technical details like test access
What is TTCN-3?

- Testing and Test Control Notation
- Internationally standardized testing language for formally defining test scenarios. Designed purely for testing
- In its essence it can be considered as a kind of scripting language that includes tons of testing specific features!

```plaintext
testcase tc_DNSquery () {
    p.send(validQuery);
    alt {
        [] p.receive(validResponse){
            setverdict( pass );}
        [else]{
            setverdict( inconc );}
    }
}
```
testcase tc_DNSquery () {
    p.send(validQuery);
    alt {
        [] p.receive(validResponse){
            setverdict( pass );
        } [else]{
            setverdict( inconc );
        }
    }
}
Main building block of TTCN-3 is a module

- Unit of compilation
- Contains definitions
- Plus optional control part

```plaintext
module DNS {

// module definitions

// module control (optional)

}
```
Module Definitions

Contains descriptions for

- What type of data the System Under Test understands
- How the System Under Tests can be accessed and what environment a test component needs
- When to communicate what with the SUT and why
- Dependencies between test cases, if any
Module Definitions (1)

- Module definitions
  - Type definitions
  - Port definitions
  - Component definitions
  - Templates
  - Test case

Control part

- Controls the execution of test cases

```plaintext
type record DNSQuery {
    charstring hostname,
    AnswerType answer optional,
    QueryType qtype
}

type union AnswerType {
    Byte ipAddress[4],
    charstring hostname
}

type integer Byte (0 .. 255);

type enumerated QueryType {
    A, NS, CNAME, MX
}
```
Module Definitions (2)

Module definitions
- Type definitions
- Port definitions
- Component definitions
- Templates
- Test case

Control part
- Controls the execution of test cases

Port definitions

```plaintext
type port DNSPort message {  
  inout DNSQuery;  
  // a port may send/receive messages  
  // of more than one type  
}
```

Component definitions

```plaintext
type component DNSTester {  
  port DNSPort P;  
  timer t := 3.0;  
  // a component may have more than one port  
}
```
Module Definitions (3)

- **Module definitions**
  - Type definitions
  - Port definitions
  - Component definitions
  - **Templates**
  - Test case

- **Control part**
  - Controls the execution of test cases

```plaintext
template DNSQuery validQuery := {
  hostname := "www.testingtech.com",
  answer := omit,
  qtype := A
}
template DNSQuery validReply modifies query := {
  answer := { ipAddress :=
    {212,227,57,158} }
}
```

```
"www.testingtech.com"  A
```

```
"www.testingtech.com"  212, 227, 57, 158  A
```
Module Definitions (4)

- Module definitions
  - Type definitions
  - Port definitions
  - Component definitions
  - Templates
  - Test case

- Control part
  - Controls the execution of test cases

```c
testcase tc_testcase1() runs on DNSTester {
    P.send(validQuery);
    P.receive(validReply);
    setverdict(pass);
}
```

// there may be more than one in a module
Module Definitions (5)

- Module definitions
  - Type definitions
  - Port definitions
  - Component definitions
  - Templates
  - Test case

Control part
- **Controls the execution of test cases**

```c
control {
    execute(tc_testcase1(), 5.0);
    while ( /* condition */ ) { };

    // more testcases might follow
    // C-like control structures available
}
```
Execution of a Test Case

testcase tc_testcase1() runs on DNSTester {
    P.send(validQuery);
    P.receive(validReply);
    setverdict(pass);
}

Is this test case definition adequate?
Dealing with Erroneous Behavior (1)

- **P.**`receive` *(validReply)* blocks until it receives a message that matches the reply
- If an unexpected message is received, any other correct message does not unblock the tester, which then blocks forever
- If no message is received, the tester will also block forever
testcase tc_testcase2() runs on DNSTester {

P.send(validQuery);

  t.start;

alt {
    [] P.receive(validReply) {
      setverdict(pass);
    }
    [] P.receive { // any message
      setverdict(fail);
    }
    [] t.timeout {
      setverdict(inconc);
    }
}

stop;
}
**Code Reusability – Altsteps and Defaults**

```plaintext
alt {
    [] P.receive(validReply) {
        setverdict(pass);
    }
    [] P.receive { // any message
        setverdict(fail);
    }
    [] t.timeout {
        setverdict(inconc);
    }
}

refactor

altstep a_RefactoredAltstep() runs on DNSTester {
    [] P.receive { // any message
        setverdict(fail);
    }
    [] t.timeout {
        setverdict(inconc);
    }
}

becomes

var default d := activate(a_RefactoredAltstep());
P.send(validQuery);
t.start;
P.receive(validReply);
setverdict(pass);
```
Non-Local DNS Query (1)

Tester

Send fully qualified hostname

Return IP address

Ask for remote DNS

Get remote address

Main Test Component

Parallel Test Component 1

Local Network Client (Client)

Parallel Test Component 2

The internet’s root name service (root NS)

Parallel Test Component 3

Remote DNS (NS)

Local Domain Name Server (DNS)

System Under Test
From Simple To Complex Test Scenarios

- Test system needs more interfaces
  - Test System Interface has to be extended

- Test behavior required at additional test interfaces
  - Behavior of Local Network Client already covered in `tc_testcase2`
  - Behavior of RootNS and NS required

- Test case that combines all parts
Parallel Test Components

- Test system interface

```c
type component DNSTester {
    port DNSPort CLIENT;
    port DNSPort ROOT;
    port DNSPort NS;
    // A component may have more than one port
}
```
Functions can be used to define the behavior of the parallel test components

```
testcase tc_testcase2() runs on DNSTester {
  var default d := activate(a_refactoredAltstep());
  P.send(validQuery);
  t.start;
  P.receive(validReply);
  setverdict(pass);
  stop;
}
```

becomes

```
function f_clientBehavior() runs on DNSTester {
  var default d := activate(a_refactoredAltstep());
  P.send(validQuery);
  t.start;
  P.receive(validReply);
  setverdict(pass);
  stop;
}
```
Additional Test Behavior

Simple „react-on-request“ behavior

```javascript
function f_rootBehavior() runs on DNSTester {
    alt {
        [] P.receive(rootQuery) {
            P.send(rootAnswer);
            setverdict(pass);
        }
        [] P.receive {
            setverdict(fail);
        }
    }
}

function f_nSBehavior() runs on DNSTester {
    alt {
        [] P.receive(nsQuery) {
            P.send(nsAnswer);
            setverdict(pass);
        }
        [] P.receive {
            setverdict(fail);
        }
    }
}
```
Additional Test Behavior

Simple „react-on-request“ behavior

```plaintext
function f_rootBehavior() runs on DNSTester {
  alt {
    [] P.receive(rootQuery) {
      P.send(rootAnswer);
      setverdict(pass);
    }
    [] P.receive {
      setverdict(fail);
    }
  }
}

function f_nSBehavior() runs on DNSTester {
  alt {
    [] P.receive(nsQuery) {
      P.send(nsAnswer);
      setverdict(pass);
    }
    [] P.receive {
      setverdict(fail);
    }
  }
}
```
Dynamic Configuration

testcase tc_testcase3() runs on MTC
system TestSystemInterface {

var DNSTester ClientComp, RootComp, NSComp;

ClientComp := DNSTester.create;
RootComp := DNSTester.create;
NSComp := DNSTester.create;

map(ClientComp:P, system:CLIENT);
map(RootComp:P, system:ROOT);
map(NSComp:P, system:NS);

RootComp.start (f_rootBehavior());
NSComp.start (f_nSBehavior());
ClientComp.start(f_clientBehavior());

ClientComp.done;
// block until ClientComp is done
stop;
}

Re-configuration during run time is possible
A Little Bit on Syntax

- Case sensitive
  - More than 146 (edition 4.5) keywords, all lower case
  - Identifiers

- Comments
  - Multi line comments: /* */
  - Single line comments: //

- Statements are terminated with: ;

- Statement blocks are enclosed in: {  }

- Operators
  - Assignment: :=
  - Comparison: ==, !=, <=, >=
Some References

The language
- www.ttcn-3.org
- de.wikipedia.org/wiki/TTCN-3
- en.wikipedia.org/wiki/TCN-3
- t-ort.etsi.org

The TTCN-3 Certificate
- www.german-testing-board.info/en/ttcn3_certificate.shtm

The Quick Reference Card
- www.blukaktus.com/card.html

Some tools
- www.ttcn-3.org/commercialtools.htm