Test Automation with TTCN-3

Introduction

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
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).

...and as mobile applications increase, mobile testing gains traction...

- 19% of businesses have fully operational Test Centers of Excellence (compared to 6% last year).
**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 -
Spending in testing (WQR 2013)

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

- Is
  - Important
  - Expensive
  - Time critical

- But
  - Only rarely practiced
  - Unsystematic
  - Performed by hand
  - Error-prone
  - Uncool ("If you are a bad programmer you might be a tester.")
  - Unconstructive
Why using TTCN-3

- Speed to Market
- High Integration
- High Reliability
- Law Changes
- Complexity
- Optimal Cost
- IP

MISMATCH

Test Methods

Requirements
Why using TTCN-3

Speed to Market

High Integration

High Reliability

Law Changes

Optimal Cost

IP

Java

TCL

C

UMTS

WLAN
Why using TTCN-3

- Speed to Market
- High Integration
- High Reliability
- Optimal Cost
- Complexity
- Law Changes
- UMTS
- WLAN
- IP

MATCH

Requirements Test Methods

High Quality
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
Standardization process

- ETSI (European Telecommunications Standards Institute) works on European telecom standards
- Divided in TCs (Technical Committees)
- TC MTS (Methods for Testing and Specification) works on the evaluation and development of methods and tools for testing and specification, helps other committees to introduce and use the new methods

Standard creation
- ETSI member submits a standard proposal to TC
- Adopting a work item for it
- Drafting of the standard
Standardization process

- Standards making process
  - Objective is to convert market needs into ETSI standards
  - Several approval procedures depending upon the deliverable type to be used
  - The process itself consists of
    - Identifying needs for standardization
    - Defining the most suitable technical committee for such standardization
    - Identification, definition, approval and adoption of work items
    - Drafting, editing and publication
History

- **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

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

- Automotive solutions
  - AUTOSAR
  - OPEN Alliance
  - AVB
  - Automotive Ethernet
  - ITS

2015
- AUTOSAR: Conformance TSP
- OPEN SIG: Alliance SIG
  Conformance TSP
- ITS-G5
- AVB: Conformance Test Suites
- 3GPP: IMS-UE

2016
- Pro AVB Test Suites
- OMA: RCS Test Suites
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
  - 2014: Edition 4.6.1
  - 2015: Edition 4.7.1

Test Automation with TTCN-3

Introduction

Basic Concepts
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!

```c
testcase tc_Hello_Bob () {
  p.send("How do you do?");
  alt {
    [] p.receive("Fine!"){
      setverdict( pass );
    }
    [else]{
      setverdict( inconc );}  //Bob asleep!
  }
}
```
TTCN-3 execution

testcase tc_Hello_Bob () {
    p.send("How do you do?");
    alt {
        [] p.receive("Fine!"){
          setverdict( pass ); }
        [else]{
          setverdict( inconc ); }
    }
}
Application areas

- Multiple application areas
  - Reactive systems with technical interfaces
  - Communication protocols
  - Software testing

- Two communication paradigms
  - Message-based communication
  - Procedure-based communication

- Different kinds of testing
  - Functional testing
  - Conformance testing
  - Scalability testing …

- Addresses the complete development cycle
  - From unit via integration testing up to system level tests
Generic protocol architecture(s)

OSI View

TCP/IP View

L7 / Application
L6 / Presentation
L5 / Session
L4 / Transport
L3 / Network
L2 / Data Link
L1 / Phy

Application Layer
Transport Layer
Internet Layer
Link Layer

HTTP, FTP, SMTP, POP, Telnet, DNS
TCP, UDP, SCTP
IPv4, IPv6
Ethernet, Token Bus, Token Ring, FDDI, IPoAC

Ethernet, Token Bus, Token Ring, FDDI, IPoAC

IPv4, IPv6
TCP, UDP, SCTP
HTTP, FTP, SMTP, POP, Telnet, DNS

Link Layer
Internet Layer
Transport Layer
Application Layer

OSI View

TCP/IP View
Generic protocol architecture(s)
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
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
  - Integration of different languages like ASN.1, XML, IDL, …
Differences TTCN-2 / TTCN-3

- **Configuration**
  - Static configuration with configuration tables
  - Dynamic configuration with arbitrary amount of components
  - Differentiation between PCOs and CPs
  - One port concept

- **Communication**
  - Asynchronous communication only
    - Abstract Service Primitives
    - Protocol Data Unit
  - Procedure and message-based communication
    - Procedures
    - Messages

- **Control**
  - Static selection of test cases via selection expression
  - Complete high level control flow mechanisms
Differences TTCN-2 / TTCN-3

- **Externalisation**
  - Test suite operations
  - External function
  - PICS / PIXIT
  - Module parameters

- **Data types, values**
  - TTCN-2 / ASN.1
  - TTCN-3, ASN.1, IDL, XML, ...

- **Modularisation**
  - Possible but seldom used
  - Central concept

- **Extensibility**
  - Not possible
  - Attributes, languages

- **Methodology**
  - Conformance Testing Methodology and Framework (CTMF), (ISO 9646/ ITU X.290)
  - No specific

- **Presentation**
  - Tabular, machine processable
  - Textual, graphical, tabular, ...

- **Implementation**
  - No runtime interfaces
  - TTCN-3 Runtime Interfaces, TTCN-3 Control Interfaces

- **Acronym**
  - Tree and Tabular Combined Notation
  - Testing and Test Control Notation
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 Support
- 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 / Spirent tools support latest edition
- Also standardized by the ITU-T as ITU-T Z.16x series
TTCN-3 by example

Local Domain Name Server

Local Network Client

Main Test Component

System Under Test

Tester

Send fully qualified hostname

Return IP address
TTCN-3 by example

Tester

Client

pass

Main Test Component

(www.testingtech.com,A)

(www.testingtech.com,212.227.57.158,A)

DNS

System Under Test
TTCN-3 modules

- Main building block of TTCN-3 is a module
  - Unit of compilation
  - Contains definitions
  - 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

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

- Control part
  - Controls the execution of test cases

```plaintext
// Type definitions

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

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

- Control part
  - Controls the execution of test cases

Port definitions

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

Component definitions

type component DNSTester {
  port DNSPort P;
  timer t := 3.0;
  // a component may have more than one port
}
Module definitions

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

- Control part
  - Controls the execution of test cases

```
module definitions

- Type definitions
- Port definitions
- Component definitions

Templates

Test case

Control part

Controls the execution of test cases

```

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

```
Module definitions

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

- Control part
  - Controls the execution of test cases

```c
module DNSTester;
    testcase tc_testcase1() runs on DNSTester {
        P.send(query);
        P.receive(reply);
        setverdict(pass);
    }

    // there may be more than one in a module
```

Client

Send
(www.testingtech.com,A)

Receive
(www.testingtech.com,212.227.57.158,A)

pass

DNS
Module definitions

- 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(query);
  P.receive(reply);
  setverdict(pass);
}

Is this test case definition adequate?

Is this an effective test case definition?
Dealing with erroneous behavior

- **P.receive** \((\text{reply})\) blocks until it receives a message that matches the reply
- If 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
Dealing with erroneous behavior

testcase tc_testcase2() runs on DNSTester
{

    P.send(query);

    t.start;

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

    stop;
}
Code reusability – Altsteps and defaults

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

becomes

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

var default d := activate(a_RefactoredAltstep());
P.send(query);
t.start;
P.receive(reply);
setverdict(pass);
Non-local DNS query

Local Domain Name Server (DNS)

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)

Main Test Component

Tester

Send fully qualified hostname

Ask for remote DNS

Get remote address

System Under Test

Ask for IP address

Return desired IP address

Return IP address
Non-local DNS query

Tester
Client

SUT
DNS

Tester
root NS

Tester
NS

("www.testingtech.com",A)

("testingtech.com",NS)

("testingtech.com", "ns.testingtech.com", NS)

("www.testingtech.com", 212.227.57.158, A)

("www.testingtech.com", 212.227.57.158, A)

("www.testingtech.com", "ns.testingtech.com", NS)

("testingtech.com", NS)

("www.testingtech.com", A)
Parallel test components

- Test system interface

```plaintext
type component TestSystemInterface {
    port DNSPort CLIENT;
    port DNSPort ROOT;
    port DNSPort NS;
}
```
From test case to behavior function

- Functions can be used to define the behavior of the parallel test components

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

becomes

```plaintext
function f_clientBehavior() runs on DNSTester {
    var default d := activate(a_refactoredAltstep());
    P.send(query);
    t.start;
    P.receive(answer);
    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);
        }
    }
}
```
Dynamic configuration

- Re-configuration during runtime is possible

```go
testcase testcase3() runs on DNSTester
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);

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

    ClientComp.done;
    // block until ClientComp is done
    stop;
}
```
A little bit on syntax

- Case sensitive
  - More than 130 (edition 4) keywords, all lower case
  - Identifiers

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

- Statements are terminated with: ;

- Statement blocks are enclosed in: { }

- Operators
  - Assignment: :=
  - Comparison: !=, ==, <=, >=
Summary

- TTCN-3 as standardized language for testing
- Easy description of test scenarios
  - Different presentation formats
- Clear specification of test configurations
  - Parallel test components / concurrent behavior
- Wide range of applicability
  - Different communication paradigms - like message based and procedure based
  - Procedure based communication covered later
Summary
Why you should use TTCN-3

- To have
  - an industrial grade
  - multi-os and
  - multi-technology

- middleware testing platform
  - to build your methods
  - to create supporting tools and tool chains
  - to be able to educate people

- in order to
  - cover the whole software development process.
Some references

- The language
  - www.ttcn-3.org
  - www.spirent.com/go/TTCN-3
  - de.wikipedia.org/wiki/TTCN-3
  - en.wikipedia.org/wiki/TTCN-3

- The TTCN-3 Certificate
  - www.german-testing-board.info/english/exam-candidates/ttcn-3-certificateR/ttcn-3R-in-a-nutshell.html

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

- Some tools
  - www.ttcn-3.org/index.php/tools/tools-com