IOT-TESTWARE – AN ECLIPSE PROJECT

Ina Schieferdecker, Sascha Kretzschmann, Michael Wagner, Axel Rennoch
QRS, Praha, Czech Republic, July 27, 2017

THE ECLIPSE PROJECT

Eclipse IoT-Testware

Background:
The open source community has produced a lot of excellent technology, frameworks and products that help with implementing IoT applications. A developer usually selects an appropriate set of technology and components and incorporates them into an IoT application framework. However, there are often gaps in the technologies available that need to be bridged. The purpose of the Eclipse IoT-Testware project is to provide an environment for the easy integration and testing of different IoT components and technologies.
OUTLINE

1. Introduction
2. IoT test language
3. TTCN-3 in use
4. FOKUS contribution to IoT testing
5. Outlook
INTRODUCTION

Where are we?

TRENDS IN IOT

<table>
<thead>
<tr>
<th>CONNECTIVITY PROTOCOLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol(s) do you use for your IoT solution?</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>TCP/IP</td>
</tr>
<tr>
<td>MQTT</td>
</tr>
<tr>
<td>CoAP</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MESSAGING STANDARDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol(s) do you use for your IoT solution?</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>XML</td>
</tr>
<tr>
<td>JSON</td>
</tr>
<tr>
<td>CoAP</td>
</tr>
<tr>
<td>Other</td>
</tr>
</tbody>
</table>

© Fraunhofer FOKUS
FURTHER ASPECTS

IoT solutions often are …
1. in harsh, unreliable environments
2. in highly dynamic configurations with large number of – typically diverse – sensors and actuators with open interfaces and
3. in resource-constrained environments

IoT test solutions need to …
− Integrate simulators for environmental conditions
− Systematically determine reference configurations
− Adjust and scale test configurations dynamically
− Be a real-time system by itself
− Support test scenarios for hybrid systems (both events and streams)

IOT TEST LANGUAGE

What do we use?
**TTCN-3 is the Testing and Test Control Notation**
- Internationally standardized testing language for formally defining test scenarios. Designed purely for testing.

```cpp
testcase Hello_Bob () {
  p.send("How do you do?");
  alt {
    []p.receive("Fine!");
    {setverdict( pass )};
    [else]
    {setverdict( inconc )}  //Bob asleep!
  }
}
```

**TTCN-3 HISTORY**
- TTCN (1992)
  - published as ISO standard
  - “Tree and Tabular Combined Notation”
  - used for protocol tests: GSM, N-ISDN, B-ISDN
- TTCN-2/2++ (1997)
  - enhancements by ETSI MTS
  - module concept, concurrency
  - used for conformance tests
TTCN-3 HISTORY (CONT.)

- TTCN-3 (2000)
- further development by ETSI MTS
- Testing and Test Control Notation
- standardised test specifications:
  - SIP, SCTP, M3UA, IPv6
  - HiperLan, HiperAccess, WiMAX
  - 3GPP LTE,
  - OMA
  - TETRA
  - MOST, AUTOSAR
  - EUROCONTROL
  - oneM2M

DESIGN PRINCIPLES OF TTCN-3

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

- Areas of Testing
  - Regression testing
  - Conformance and functional testing
  - Interoperability and integration testing
  - Real-time, performance, load and stress testing
  - Security testing

- Used for system and product qualification and certification, e.g. for GCF/PTCRB certification of handsets
TTCN-3 IS DESIGNED FOR DYNAMIC TESTING

TTCN-3 Test Case

Port.send(Stimulus)  Port.receive(Response)

• Assignment of a Test Verdict

System Under Test

MAJOR LANGUAGE ELEMENTS OF TTCN-3 NOTATION

<table>
<thead>
<tr>
<th>module definitions</th>
<th>Imports</th>
<th>Importing definitions from other modules defined in TTCN-3 or other languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Types</td>
<td>User defined data types (messages, PDUs, information elements, …)</td>
<td></td>
</tr>
<tr>
<td>Test Data</td>
<td>Test data transmitted/expected during test execution (templates, values)</td>
<td></td>
</tr>
<tr>
<td>Test Configuration</td>
<td>Definition of the test components and communication ports</td>
<td></td>
</tr>
<tr>
<td>Test Behavior</td>
<td>Specification of the dynamic test behavior</td>
<td></td>
</tr>
</tbody>
</table>
A TTCN-3 TEST SYSTEM

Test System User

- TE: TTCN-3 Executable
- TM: Test Management
- TL: Test Logging
- CD: Codec
- CH: Component Handling
- SA: System Adapter
- PA: Platform Adapter
- SUT: System Under Test

ETSI ES 201 873-1 TTCN-3 Core Language (CL)
ETSI ES 201 873-5 TTCN-3 Runtime Interface (TRI)
ETSI ES 201 873-6 TTCN-3 Control Interfaces (TCI)

IMPLEMENTATION

Test Specification

Test System

Automated Test Execution and Reporting
TTCN-3 TECHNOLOGY OVERVIEW

Language mappings
- ASN.1
- IDL
- XSD
- JSON

Extensions
- Documentation
- Advanced parameterization
- Behaviour types
- Static configuration
- Real-time support
- Continuous signal support
- Advanced TRI
- Advanced Matching

Core language
- TTCN-3 Structuring: Imports, Groups, Attributes
- TTCN-3 Behaviour
- TTCN-3 Data

TRI/TCI mapping
- Java
- C
- C++
- XML
- C#

How do we use it?

TTCN-3 IN USE
ELEVATOR DEMO CONFIGURATION

Dashboard

Broker (SUT)

Cloud

Gateway

Elevator

PLC

Tester

TTCN-3

ELEVATOR DEMO CONFIGURATION (CONT.)
MODEL-BASED FUZZING

**Challenge:** Finding 0-day vulnerabilities in a highly automated, efficient manner

**Solution:** Model-based Fuzzing
- Aims at fault input validation
- Stressing the SUT with semi-valid inputs

---

MODEL-BASED FUZZING

FUZZING TOOL

- Supports generation and mutation based fuzzing
- Platform-independent: is implemented in Java
- Language-independent: provides an XML-based interface
- Automated: automatically selects appropriate fuzzing heuristics
- Efficient & scalable: the user can decide which fuzzing heuristics shall be used
- Amount of fuzz test data specifiable: avoids generating billions of values

https://github.com/fraunhoferfokus/Fuzzino
EXECUTED TEST PROCESS

1. Provide the devices
2. Identify the used technologies
3. Develop the tests
4. Build the test setup
5. Build multiple test setups
6. Run the tests long-term
7. Deduct conclusions
8. Narrow down tests specific to the device
9. Re-run the tests

EXECUTION (VIDEO)

detecting vulnerabilities using fuzzing
EXECUTION

Invalid latitude

FOKUS CONTRIBUTION TO IOT TESTING

What else?
TESTLAB (TESTING AND CERTIFICATION)

- Focussing on open source tools (Eclipse)
- Creating test suites for IoT protocols (MQTT, CoAP, …)
- Providing several end devices
- Supporting different test configurations
- “Come in and test”

- Future certification
  - “Light weight” selection of criteria
  - “Self certification” if tests are successful

ECLIPSE IOT TESTWARE

- Approved by Eclipse Foundation: https://projects.eclipse.org/proposals/eclipse-iot-testware
- Creation of TTCN-3 test suites for CoAP and MQTT
- Project partners: relayr GmbH, Ericsson, LAAS/CNRS, itemis AG, Spirent Communications, Easy Global Market

- Current schedule
  - 2017Q2: creation of a catalogue for test objectives (test purposes)
  - 2017Q3: initial publication of implemented TTCN-3 tests
TEST CONFIGURATIONS

CoAP

MQTT

THE TEST EXECUTION TOOL
IOT-T PROJECT

Customers:
- Requirements and Applications
- Testing and Certification Pilots

New Requirements

Methods and Tools
- Fraunhofer IPK
- Fraunhofer FOKUS

Tests Results and Certificates
- relayr
- DEKRA

IoT Testware
- eclipse
- All project partners

http://www.iot-t.de/en/
What are further ideas?

- Two advanced IoT testing approaches:
  - Virtualized testing (with TTCN-3)
  - TTCN-3 virtualized

- Both could provide advantages for IoT testing:
  - flexibility with test configurations
  - create test suites faster
  - run tests even "on" constrained devices
  - ...

© Fraunhofer FOKUS
**TTCN-3 VIRTUALIZED**

- Easy solution to write your test cases “online”
- Deploy your test suite (Java, C++ or as service)
- Run the executables

+ Hide complexity ➔ everyone can write tests
+ Test implementation is straight forward
- Tests may not running on highly constrained devices
- Still difficult to configure other parts of the test system

**VIRTUALIZED TESTING**

<<service>>

- TTCN-3 compiler
- MQTT test suite
- CoAP test suite
- DB
- Security tools
- Adapter or delegator layer
- Monitor

<<call>>

- CoAP Endpoint
- MQTT Broker

<<stimulate>>
PROS AND CONS OF VIRTUALIZED TESTING

+ Hide complexity → “come in and test”
+ Extensible → add new testing tools, test suites, …
+ Handle different dynamic configurations
+ Simplify testing against highly constrained devices
- Are we sure that we can test “everything”?  
- Complex technical and architectural challenges

VIRTUALIZED TESTING WITH NODE-RED?

Virtualized test component created in NODE-RED
Thank you for your attention!

www.fokus.fraunhofer.de
(System Quality Center)