




# IOT-TESTWARE – AN ECLIPSE PROJECT

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





# THE ECLIPSE PROJECT

 Create account  Log in




[GETTING STARTED](#) [MEMBERS](#) [PROJECTS](#) [MORE ▾](#)

Google Custom Search 

 **DOWNLOAD**

[HOME](#) / [PROJECTS](#) / [TECHNOLOGY PROJECT](#) / [ECLIPSE IOT-TESTWARE](#) / [ECLIPSE IOT-TESTWARE](#)

This proposal has been approved and the **Eclipse IoT-Testware** project has been created. 

## Eclipse IoT-Testware

### BASICS

This proposal is in the Project Proposal Phase (as defined in the **Eclipse Development Process**) and is written to declare its intent and scope. We solicit additional participation and input from the community. Please login and add your feedback in the comments section.

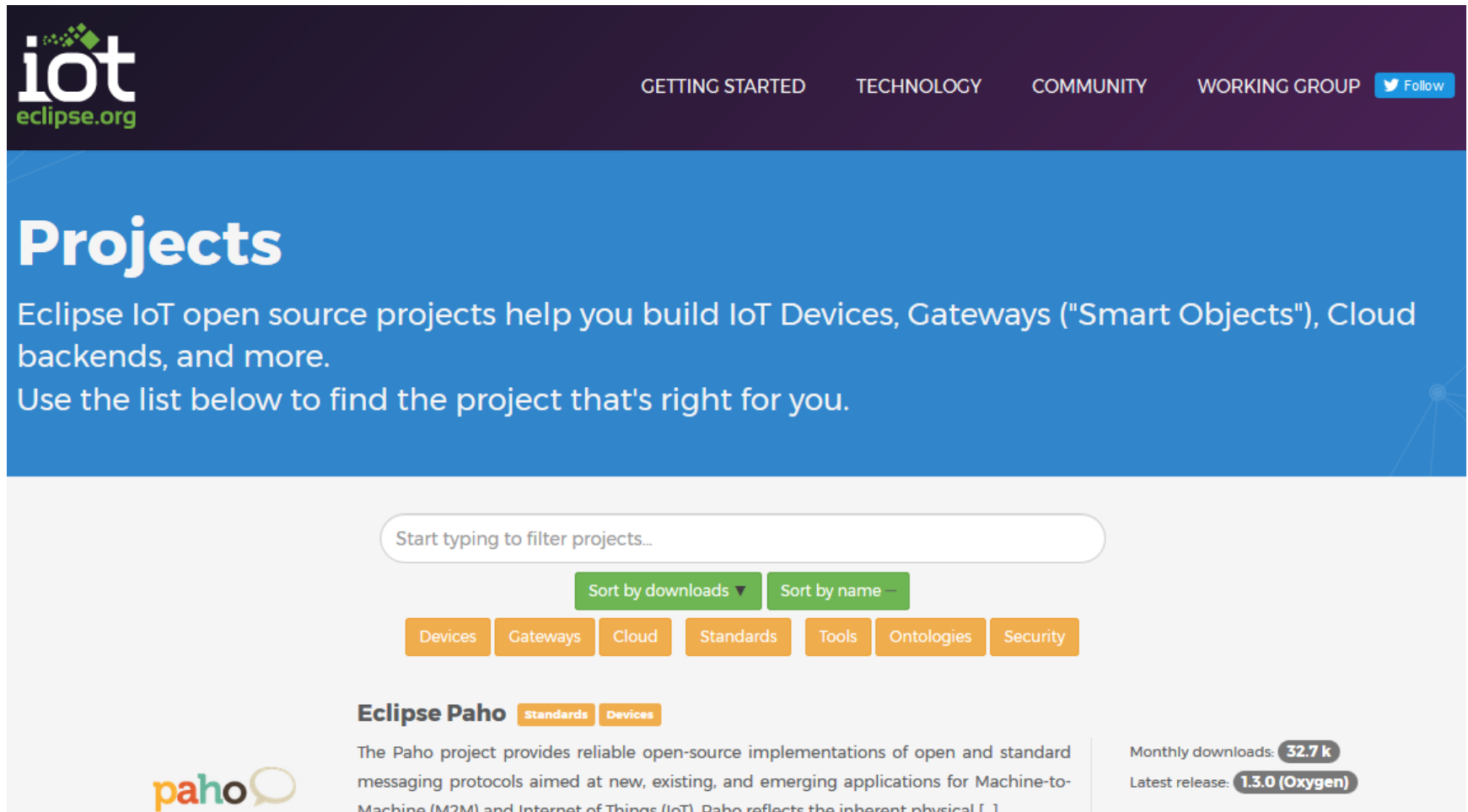
**Parent Project:**  
**Technology Project**

### 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 application. The chosen components need to support the implementation of all relevant aspects of an IoT solution including:



# THE CONTEXT



The screenshot shows the Eclipse IoT Projects website. At the top is a dark purple navigation bar with the 'iot eclipse.org' logo on the left and links for 'GETTING STARTED', 'TECHNOLOGY', 'COMMUNITY', 'WORKING GROUP', and a 'Follow' button with a Twitter icon. Below the navigation bar is a large blue banner with the word 'Projects' in white. Underneath the banner, text explains that Eclipse IoT open source projects help build IoT Devices, Gateways ('Smart Objects'), Cloud backends, and more, and encourages users to find the right project. A search bar with the placeholder 'Start typing to filter projects...' is followed by two green buttons: 'Sort by downloads' and 'Sort by name'. Below these are seven orange category buttons: 'Devices', 'Gateways', 'Cloud', 'Standards', 'Tools', 'Ontologies', and 'Security'. The 'Eclipse Paho' project is highlighted, with 'Standards' and 'Devices' tags. To the left of the project name is the Paho logo. The project description states it provides reliable open-source implementations of open and standard messaging protocols for Machine-to-Machine (M2M) and Internet of Things (IoT). To the right, it shows 'Monthly downloads: 32.7 k' and 'Latest release: 1.3.0 (Oxygen)'.

**iot**  
eclipse.org

GETTING STARTED TECHNOLOGY COMMUNITY WORKING GROUP [Follow](#)

## Projects


Eclipse IoT open source projects help you build IoT Devices, Gateways ("Smart Objects"), Cloud backends, and more.  
Use the list below to find the project that's right for you.

Start typing to filter projects...

Sort by downloads ▼ Sort by name —

Devices Gateways Cloud Standards Tools Ontologies Security

**Eclipse Paho** Standards Devices

 The Paho project provides reliable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT). Paho reflects the inherent physical [ ]

Monthly downloads: **32.7 k**  
Latest release: **1.3.0 (Oxygen)**



# OUTLINE

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

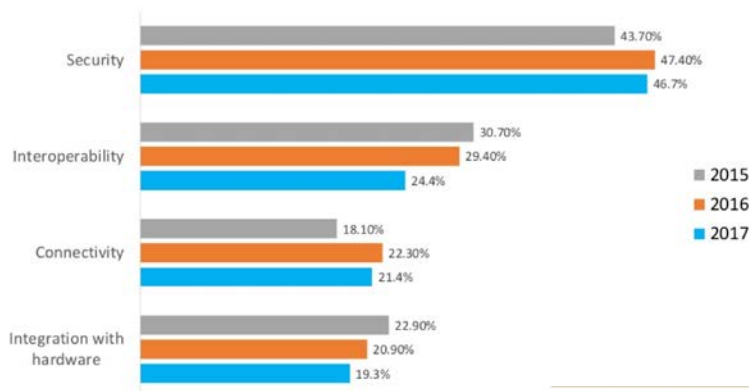
# INTRODUCTION

**Where are we?**



# TRENDS IN IOT

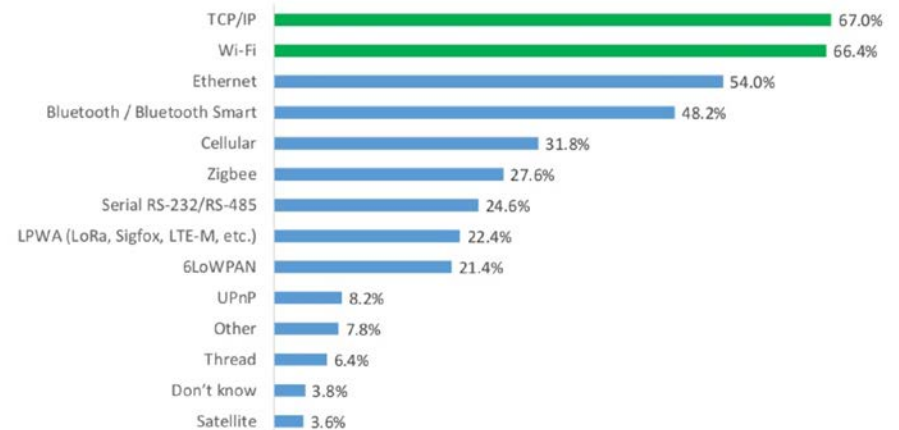
## TOP IOT CONCERNS / TRENDS 2015-2017



IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc.

## CONNECTIVITY PROTOCOLS

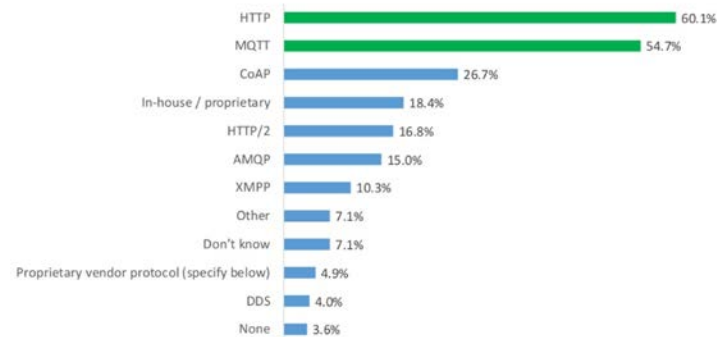
*What connectivity protocol(s) do you use for your IoT solution?*



IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc.

## MESSAGING STANDARDS

*What messaging protocol(s) do you use for your IoT solution?*



IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc.

# REFERENCE MODEL (ONE OF MANY)

## IOT PRINCIPAL COMMUNICATION ARCHITECTURE

### APPLICATION LEVEL

Endpoints and Applications  
(User interfaces and access)

Processes  
(Collaboration and business processes)

Services  
(Reporting, command and control)

*Control center and cockpits*

*Service and application frameworks*

### PLATFORM LEVEL

Data Analytics and Visualization  
(Aggregation, mash ups, etc.)

*Remote computation  
(learning, constraint solving, rule engines, decision management, etc.)*

Data Storage  
(Accumulation)

*Also data from other sources  
incl. open data*

### NETWORK LEVEL

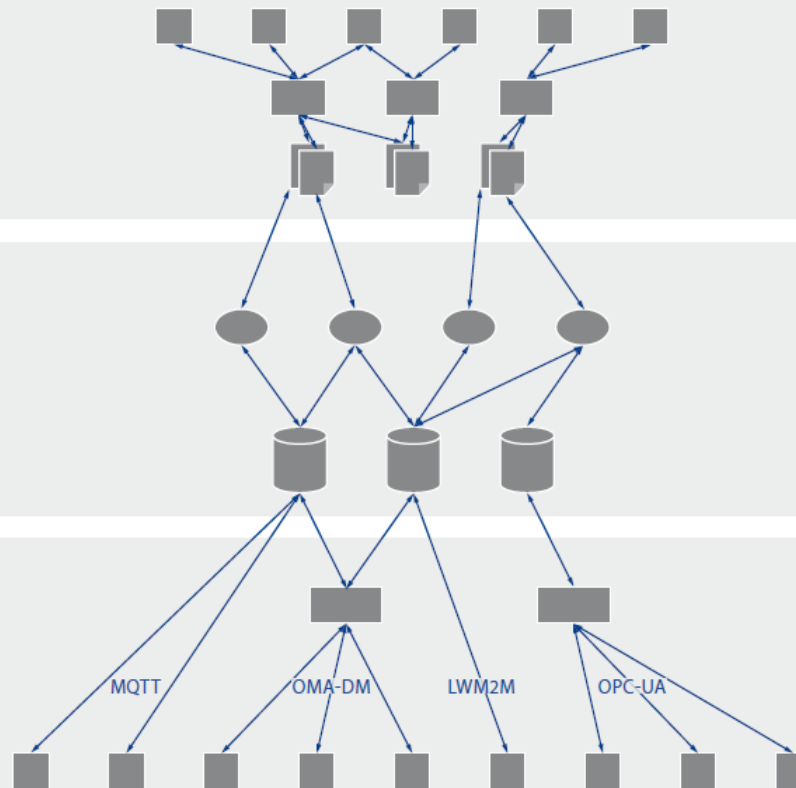
Edge Computing  
(Node data analysis)

*Local computation*

Node Connectivity  
(Interoperable, heterogenous)

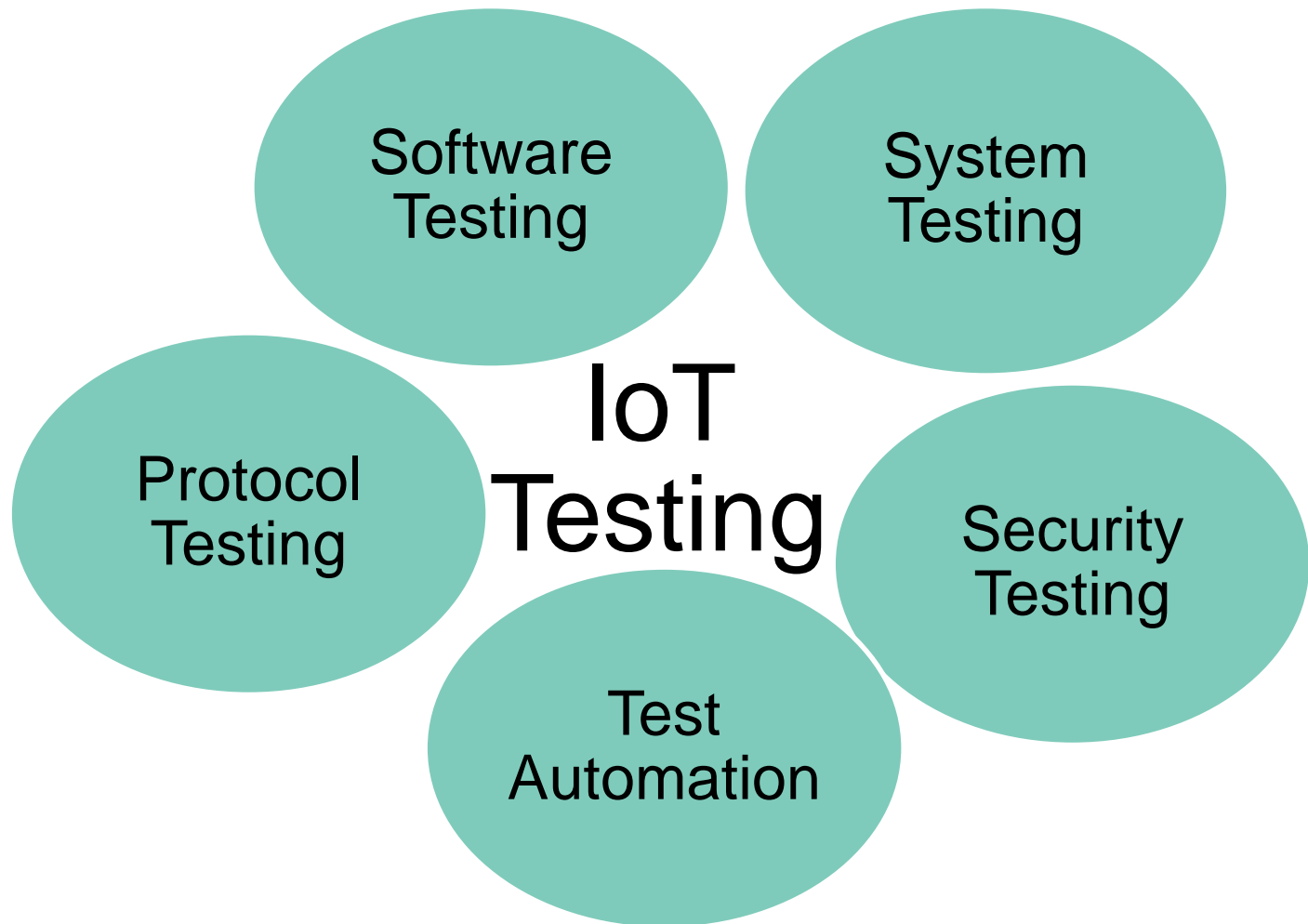
*Also CoAP, HTTP, or proprietary*

Edge Nodes (Intelligent, of all types –  
sensors, devices, machines)





# INTEGRATION OF SEVERAL TESTING APPROACHES



# 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?**

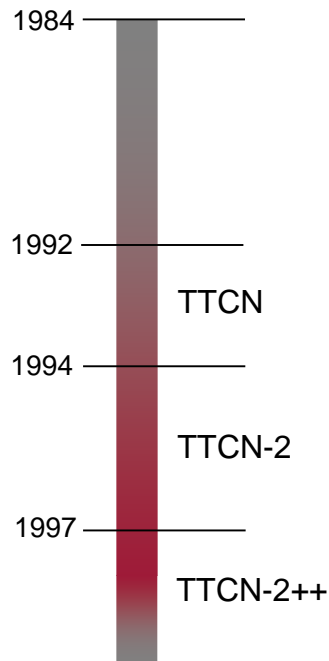
# CHALLENGE TEST AUTOMATION

- TTCN-3 is the Testing and Test Control Notation
- Internationally standardized testing language for formally defining test scenarios. Designed purely for testing



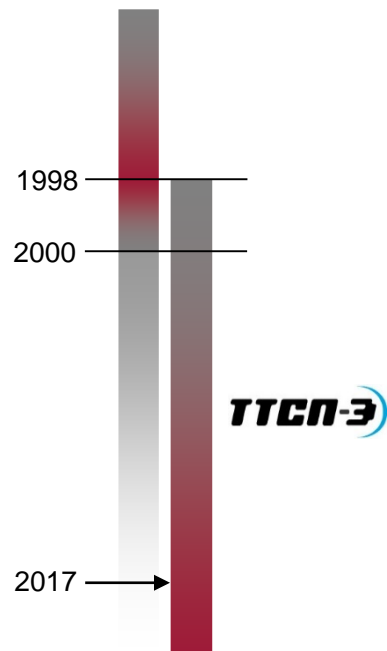
```
testcase Hello_Bob () {  
    p.send("How do you do?");  
    alt {  
        [!p.receive("Fine!");  
            {setverdict( pass )};  
        [else]  
            {setverdict( inconc )} //Bob asleep!  
    }  
}
```

# 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.)

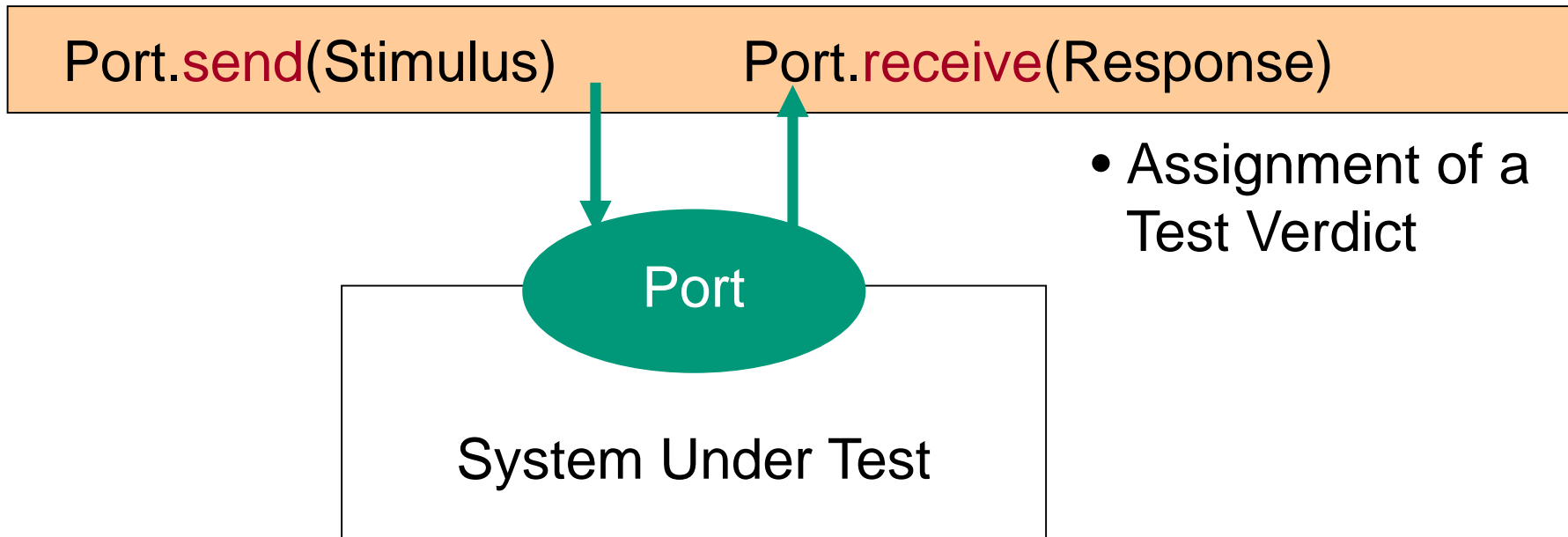


- TTCN-3 (2000)
- further development by ETSI MTS
- Testing and Test Control Notation
- standardised test specifications:
  - SIP, SCTP, M3UA, IPv6
  - HiperLan, HiperAccess, WiMAX
  - 3GPP LTE,
  - OMA
  - TETRA
  - MOST, AUTOSAR
  - EUROCONTROL
  - **oneM2M**

- **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

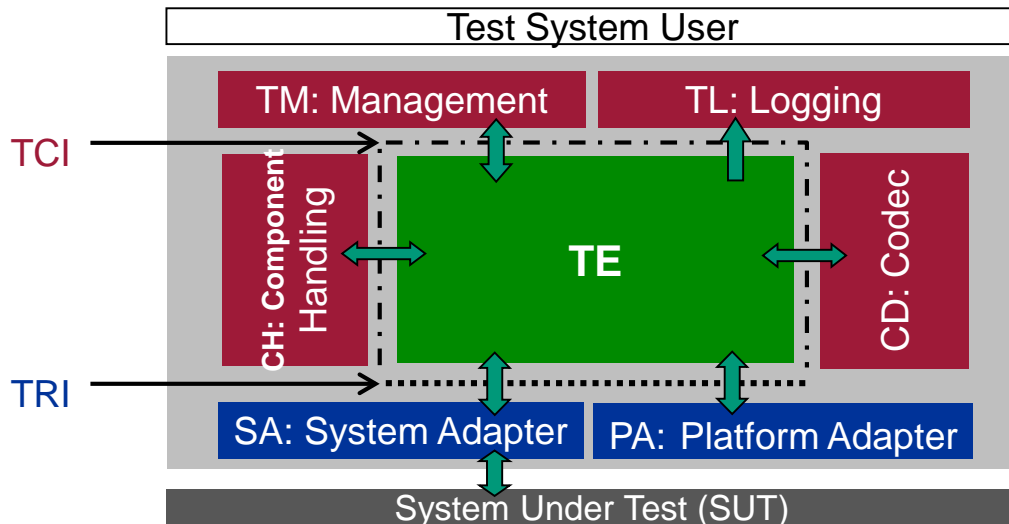


# MAJOR LANGUAGE ELEMENTS OF TTCN-3 NOTATION

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



# A TTCN-3 TEST SYSTEM

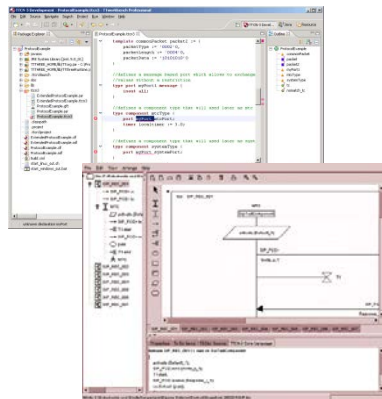


- TE – TTCN-3 Executable
- TM – Test Management
- TL – Test Logging
- CD – Codec
- CH – Component Handling
- SA – System Adapter
- PA – Platform Adapter
- SUT – System Under Test

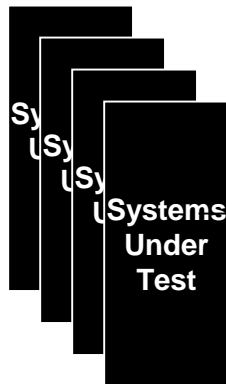
ETSI ES 201 873-1 TTCN-3 Core Language (CL)  
ETSI ES 201 873-5 TTCN-3 Runtime Interface (TRI)  
ETSI ES 201 873-6 TTCN-3 Control Interfaces (TCI)

# IMPLEMENTATION

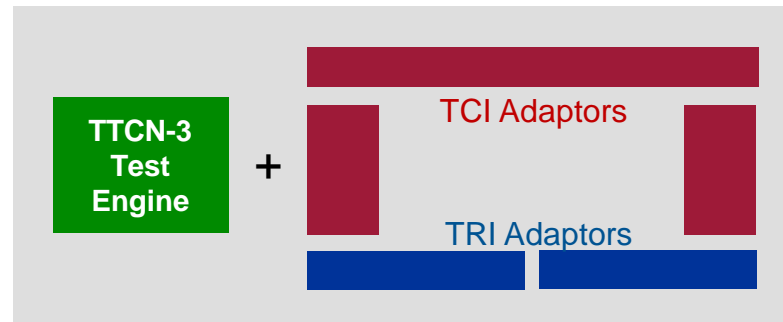
## Test Specification



**TTCN-3**

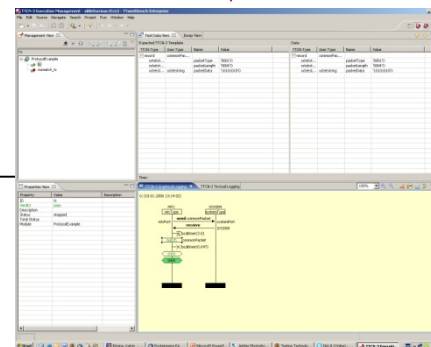


## Test System

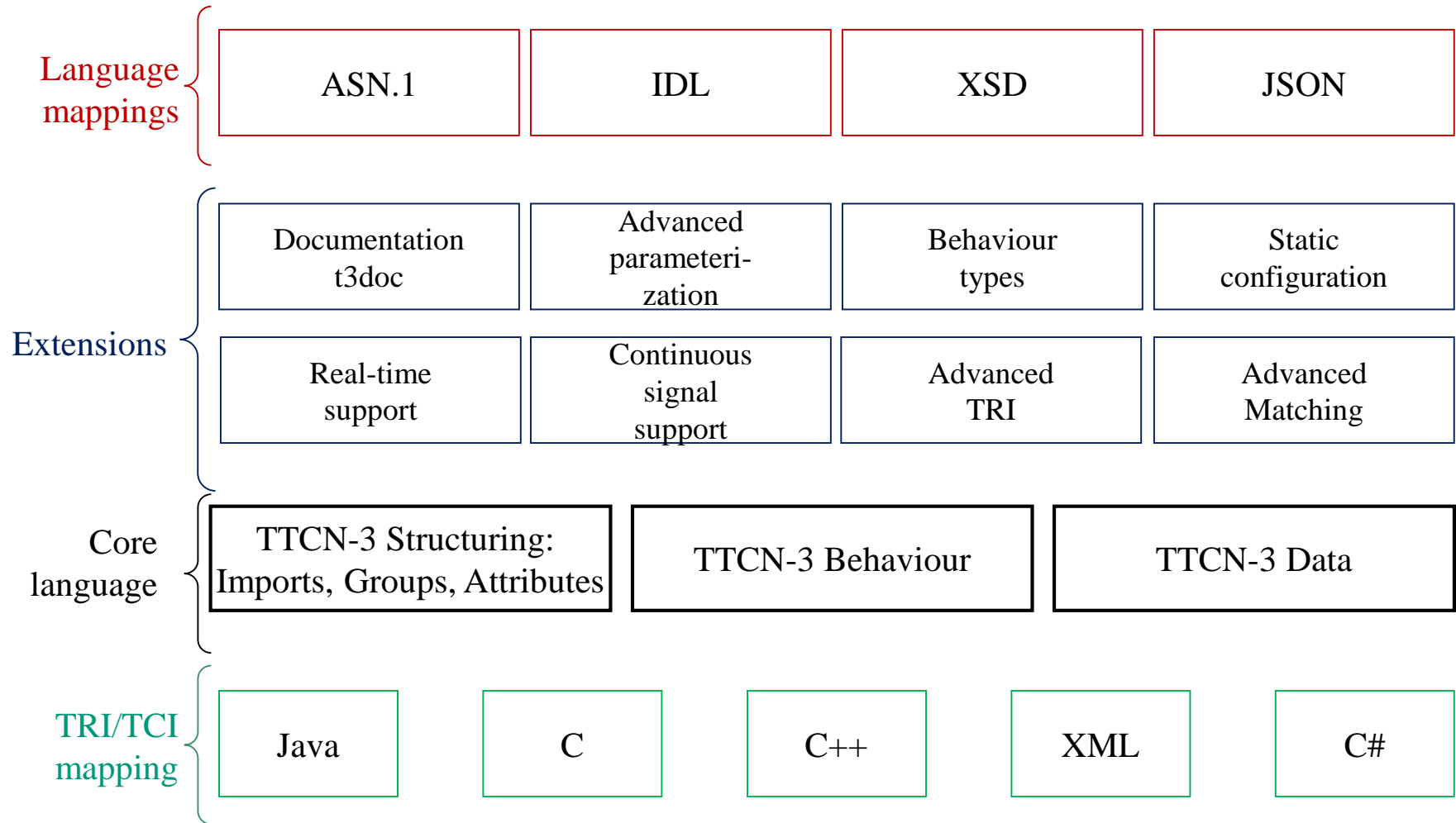


Communication /  
Invocation

## Automated Test Execution and Reporting



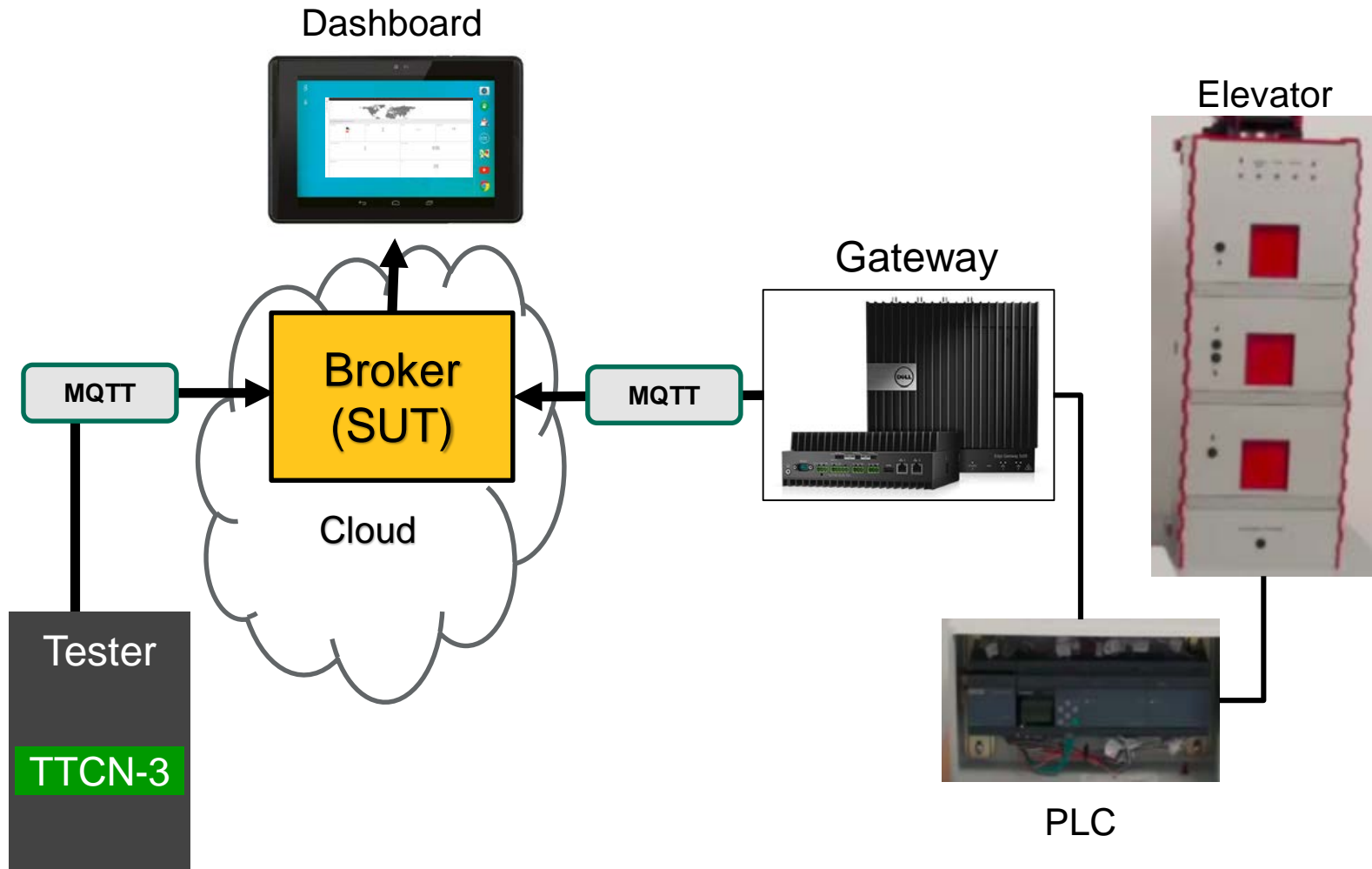
# TTCN-3 TECHNOLOGY OVERVIEW



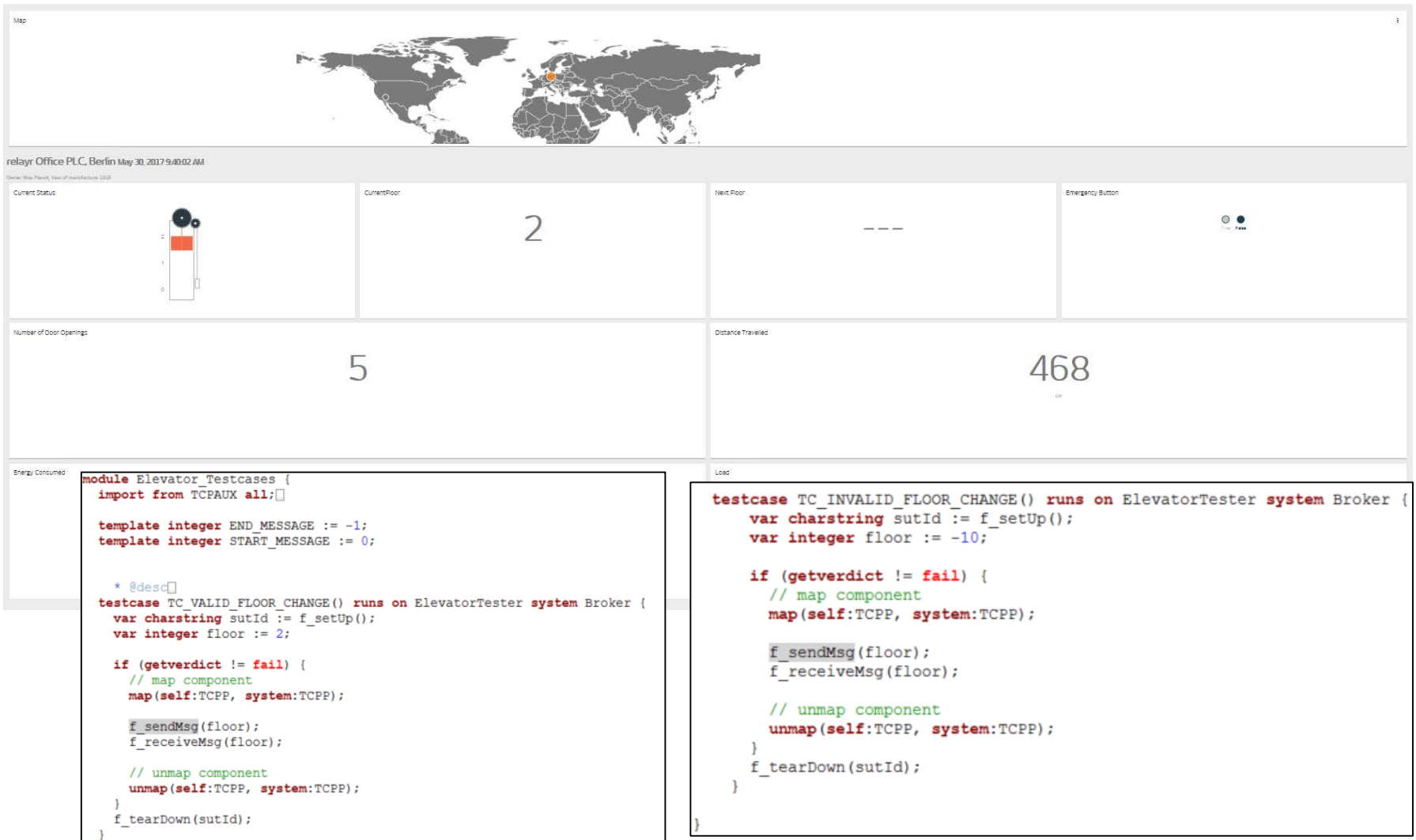
# TTCN-3 IN USE

**How do we use it?**

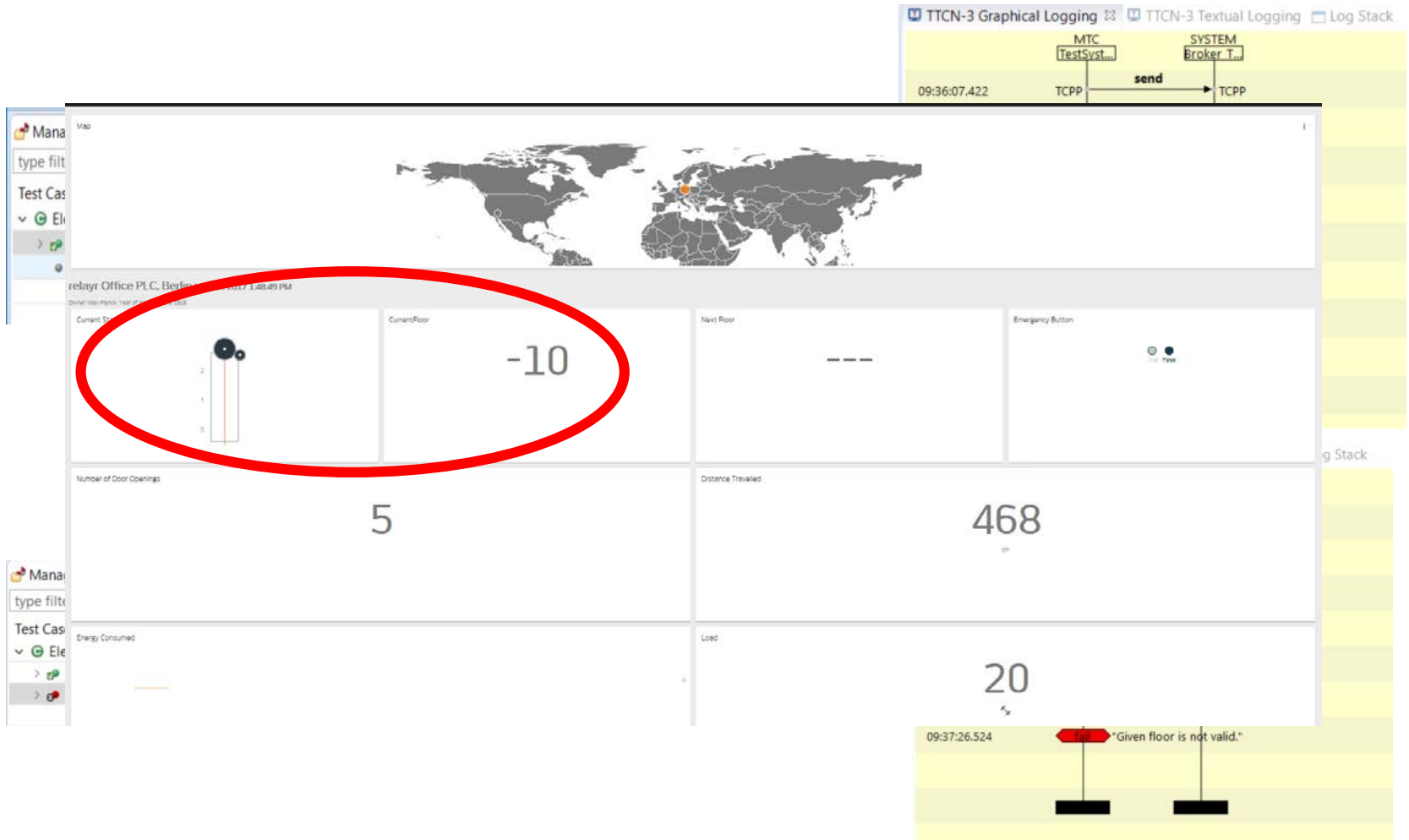
# ELEVATOR DEMO CONFIGURATION



# ELEVATOR DEMO CONFIGURATION (CONT.)



# ELEVATOR DEMO CONFIGURATION (CONT.)

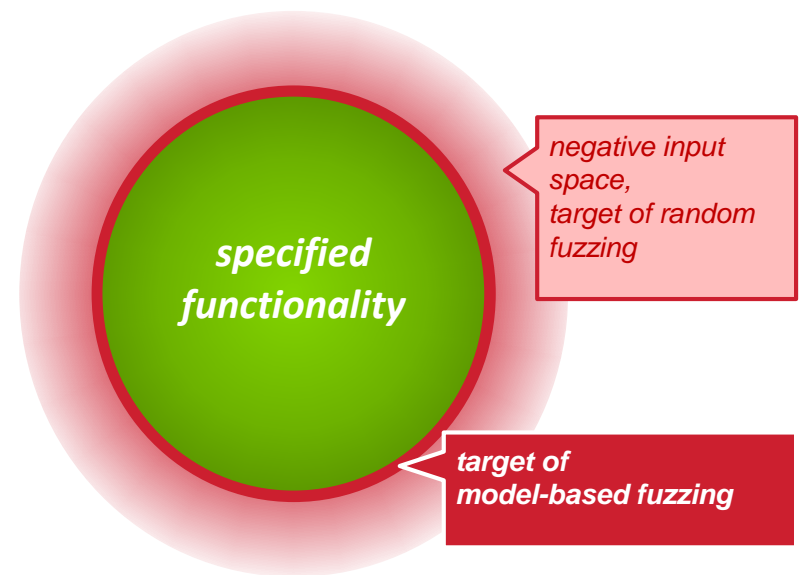


# MODEL-BASED FUZZING

**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

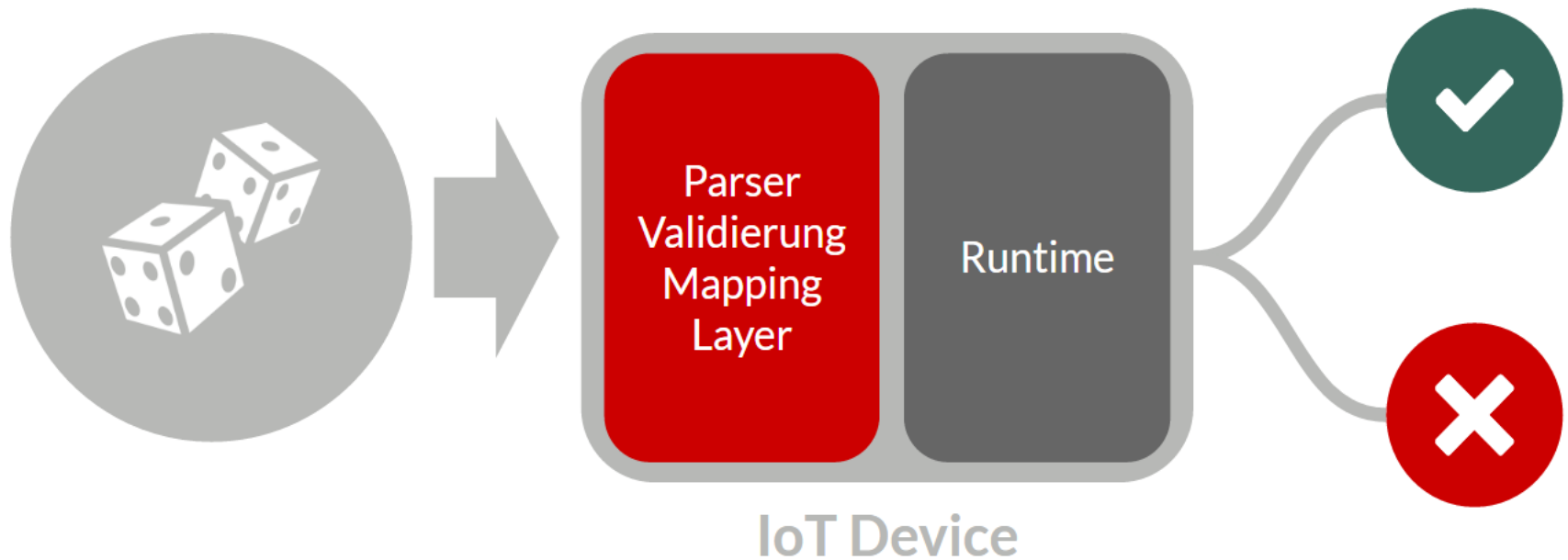


see also:

Takanen, Ari; DeMott, Jared D.; Miller, Charles: *Fuzzing for Software Security Testing and Quality Assurance*, 2008 ; ISBN 978-1-59693-214-2



# MODEL-BASED FUZZING



# FUZZING TOOL



FUZZINO

<https://github.com/fraunhoferfokus/Fuzzino>

- 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

# 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)



# 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



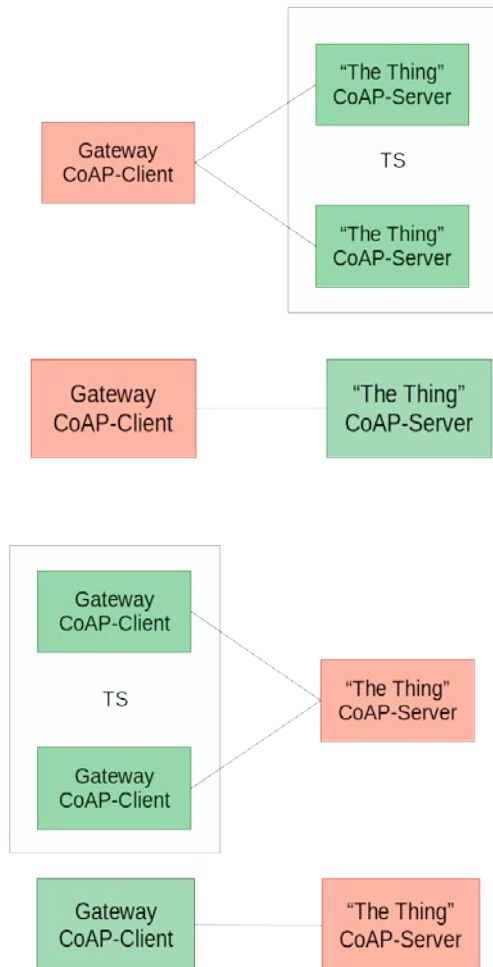
# ECLIPSE IOT TESTWARE



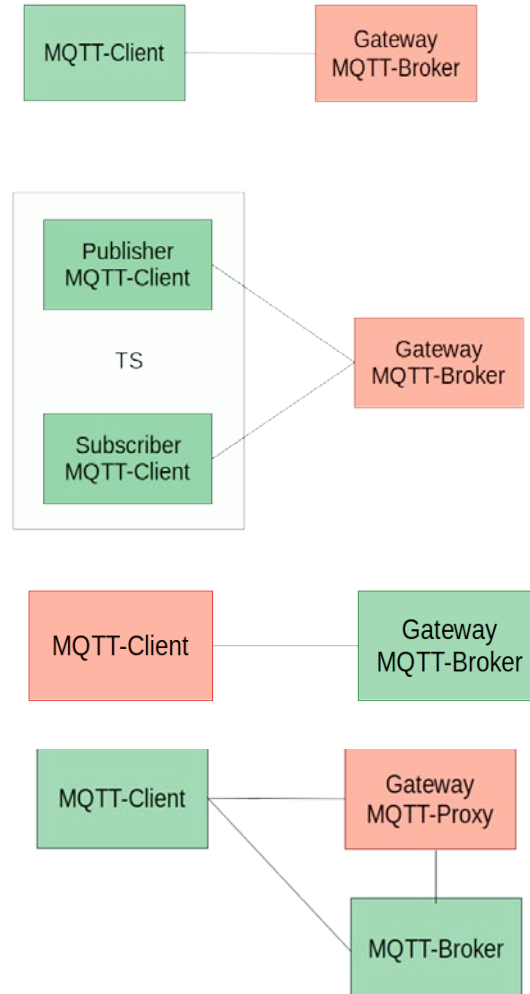
- Approved by Eclipse Foundation :  
<https://projects.eclipse.org/proposals/eclipse-iot-testware>
- Creation of TTCN-3 test suites for CoAP and MQTT
- Project partners: relayr GmbH, Ericsson, LAAS/CNRS, itemis AG, Spirent Communications, Easy Global Market
- Current schedule
  - 2017Q2: creation of a catalogue for test objectives (test purposes)
  - 2017Q3: initial publication of implemented TTCN-3 tests

# TEST CONFIGURATIONS

## CoAP



## MQTT





# THE TEST EXECUTION TOOL

Create account Log in



Google Custom Search

DONATE

GETTING STARTED MEMBERS PROJECTS MORE

HOME / PROJECTS / TOOLS PROJECT / ECLIPSE TITAN

## Eclipse Titan

Overview Downloads Who's Involved Developer Resources Governance Contact Us

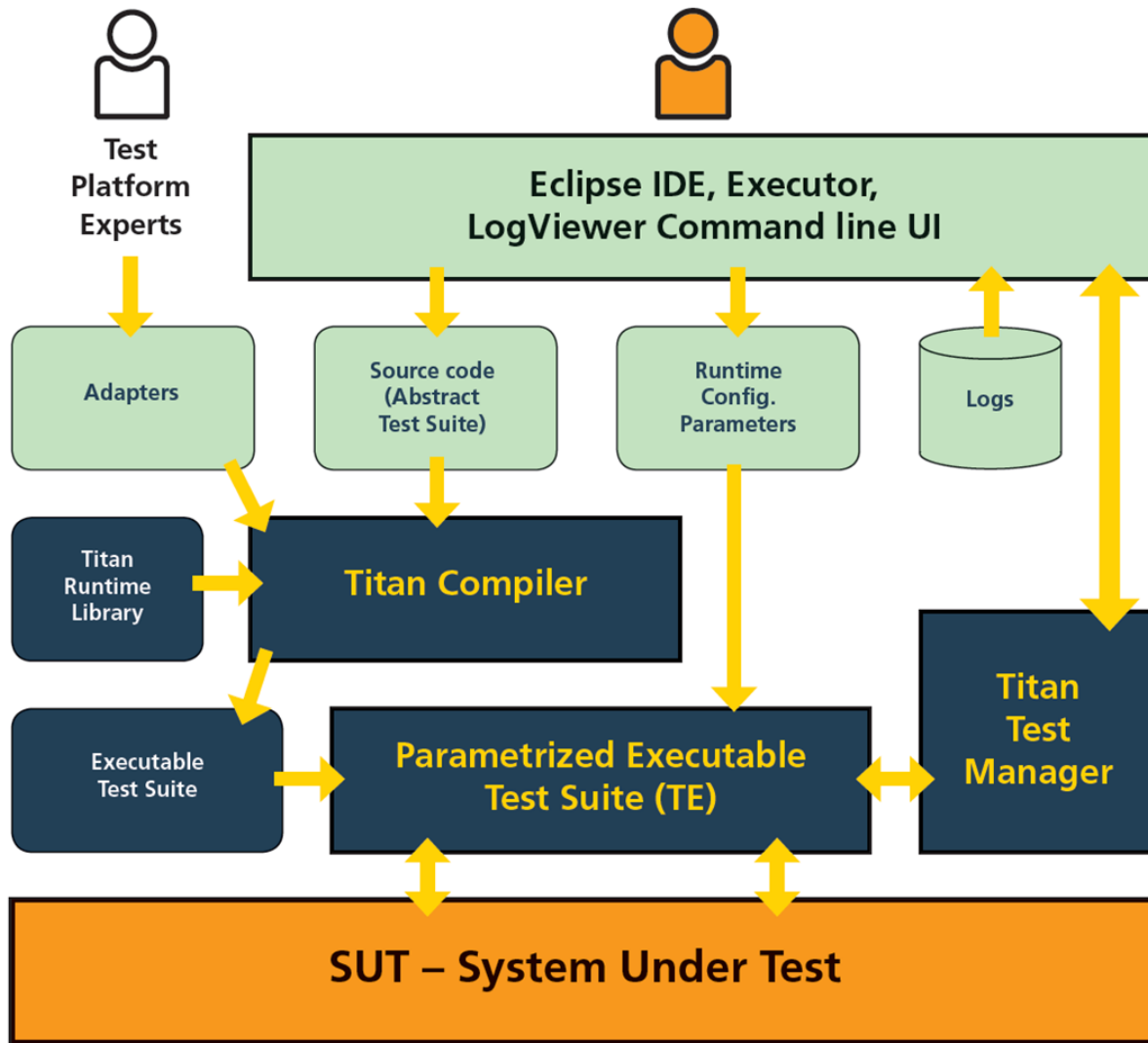
Titan is a TTCN-3 compilation and execution environment with an Eclipse-based IDE. TTCN-3 is a modular language specifically designed for testing (the acronym itself stands for Test and Test Conformance Notation), standardized by ETSI (see [www.ttcn-3.org](http://www.ttcn-3.org)) and endorsed by ITU. The user of the tool can develop test cases, test execution logic and build the executable test suite for several platforms. Titan consists of a core part, executing in a Unix/Linux-like environment and a set of Eclipse plug-ins.

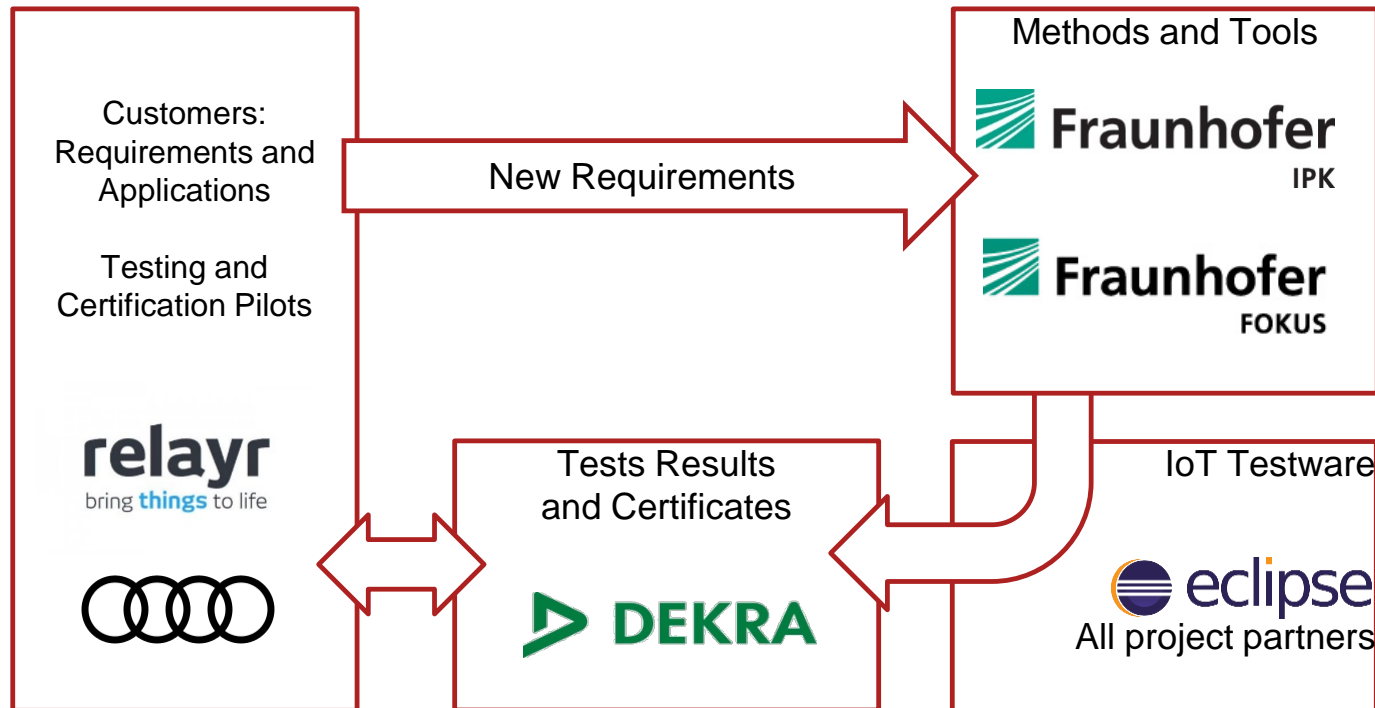
### Titan



Eclipse IDE, Executor, Test Platform







<http://www.iot-t.de/en/>

# OUTLOOK

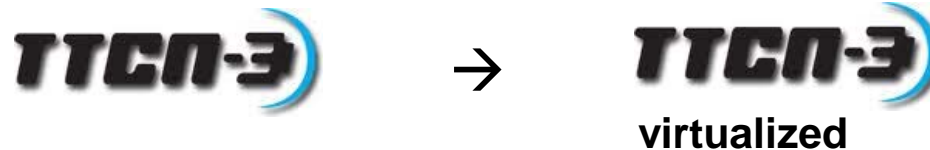
**What are further ideas?**

# OUTLOOK

- 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
  - ...



# TTCN-3 VIRTUALIZED

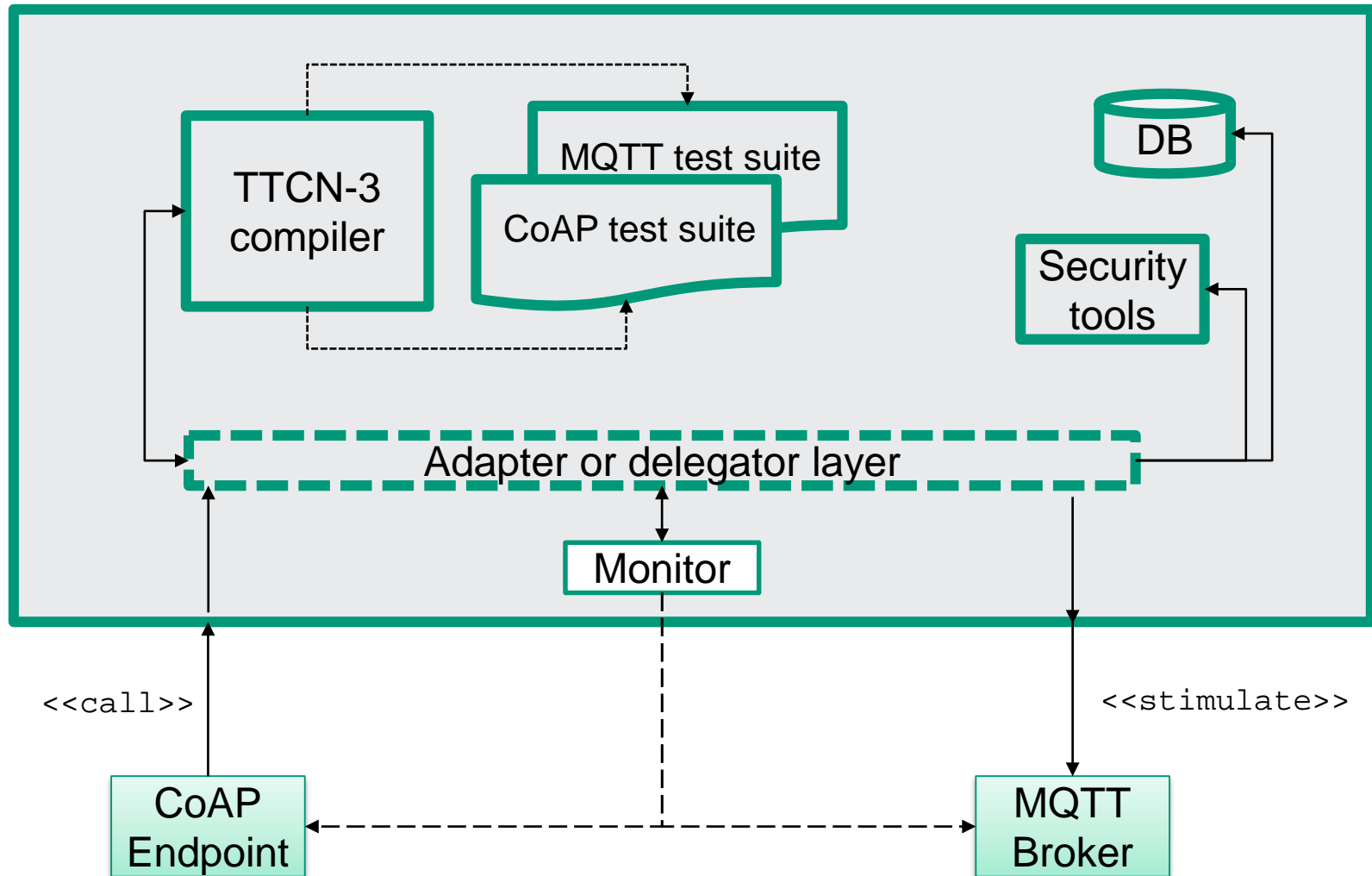


- Easy solution to write your test cases “online“
  - Deploy your test suite (Java, C++ or as service)
  - Run the executables
- 
- + Hide complexity → everyone can write tests
  - + Test implementation is straight forward
  - Tests may not running on highly constrained devices
  - Still difficult to configure other parts of the test system



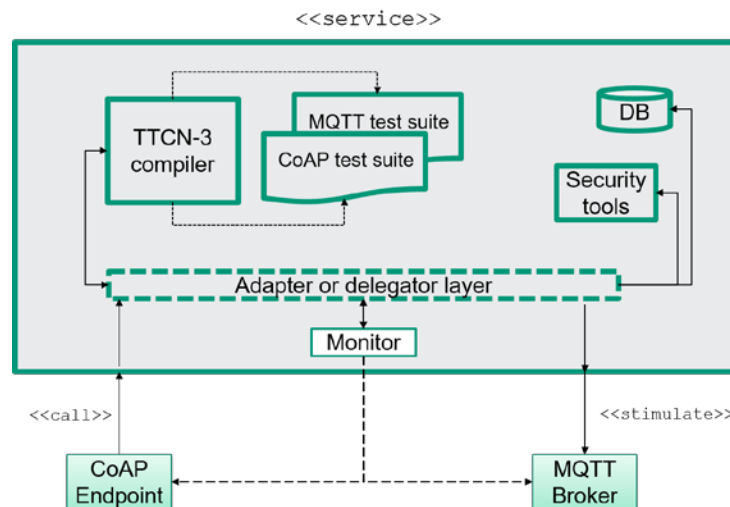
# VIRTUALIZED TESTING

<<service>>



# PROS AND CONS OF VIRTUALIZED TESTING

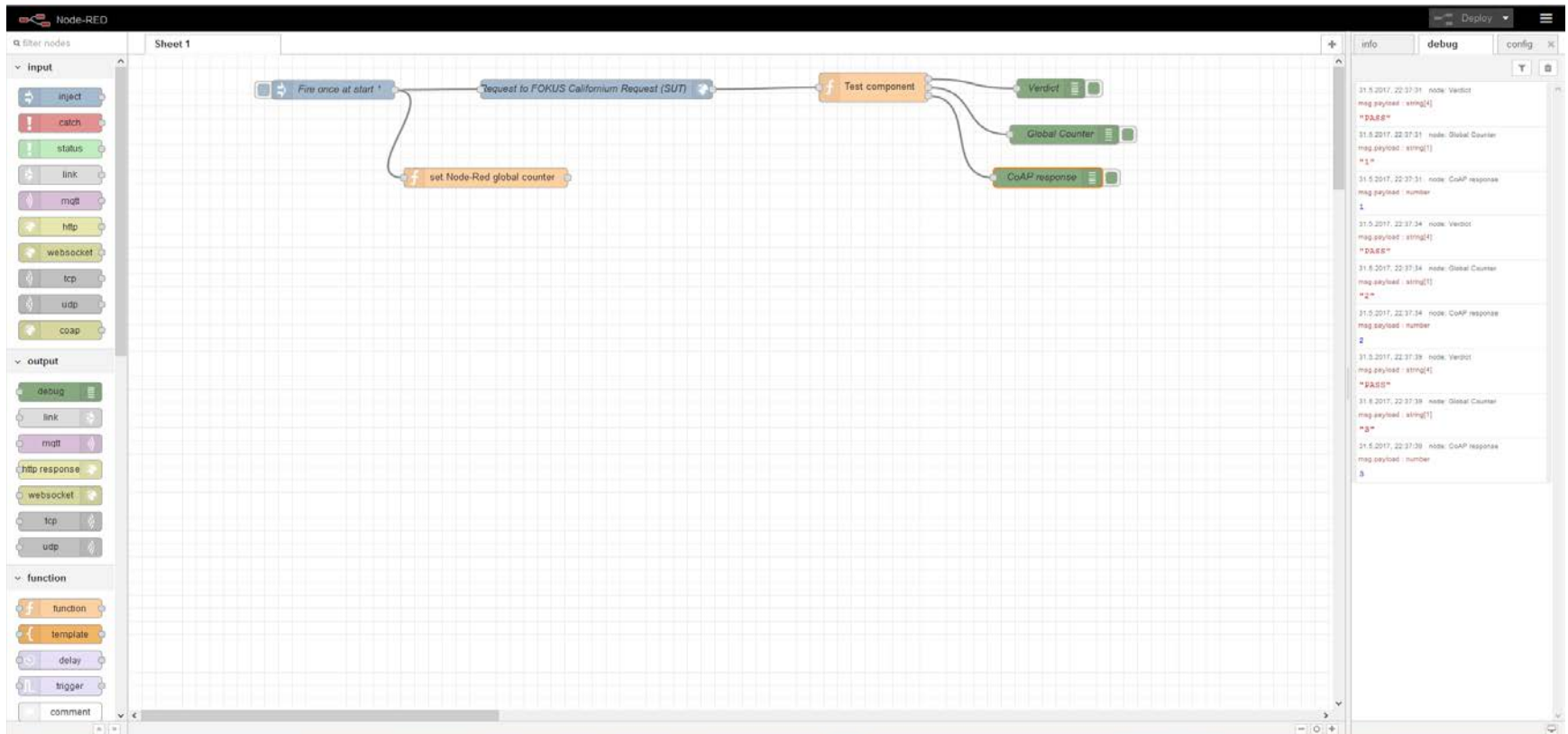
- + Hide complexity → “come in and test“
- + Extensible → add new testing tools, test suites, ...
- + Handle different dynamic configurations
- + Simplify testing against highly constrained devices
- Are we sure that we can test “everything“ ?
- Complex technical and architectural challenges





# VIRTUALIZED TESTING WITH NODE-RED?

## Virtualized test component created in NODE-RED



**Thank you  
for your attention!**

**[www.fokus.fraunhofer.de](http://www.fokus.fraunhofer.de)**  
**(System Quality Center)**

# CONTACTS

Fraunhofer FOKUS  
Kaiserin-Augusta-Allee 31  
10589 Berlin, Germany  
[www.fokus.fraunhofer.de](http://www.fokus.fraunhofer.de)

Ina Schieferdecker, Michael Wagner, Axel Rennoch & Sascha Kretzschmann  
{ina.schieferdecker, michael.wagner, axel.rennoch,  
sascha.kretzschmann}@fokus.fraunhofer.de  
Phone +49 30 3463-7201