IOT-TESTWARE – AN ECLIPSE PROJECT

© Vadim Makhorov

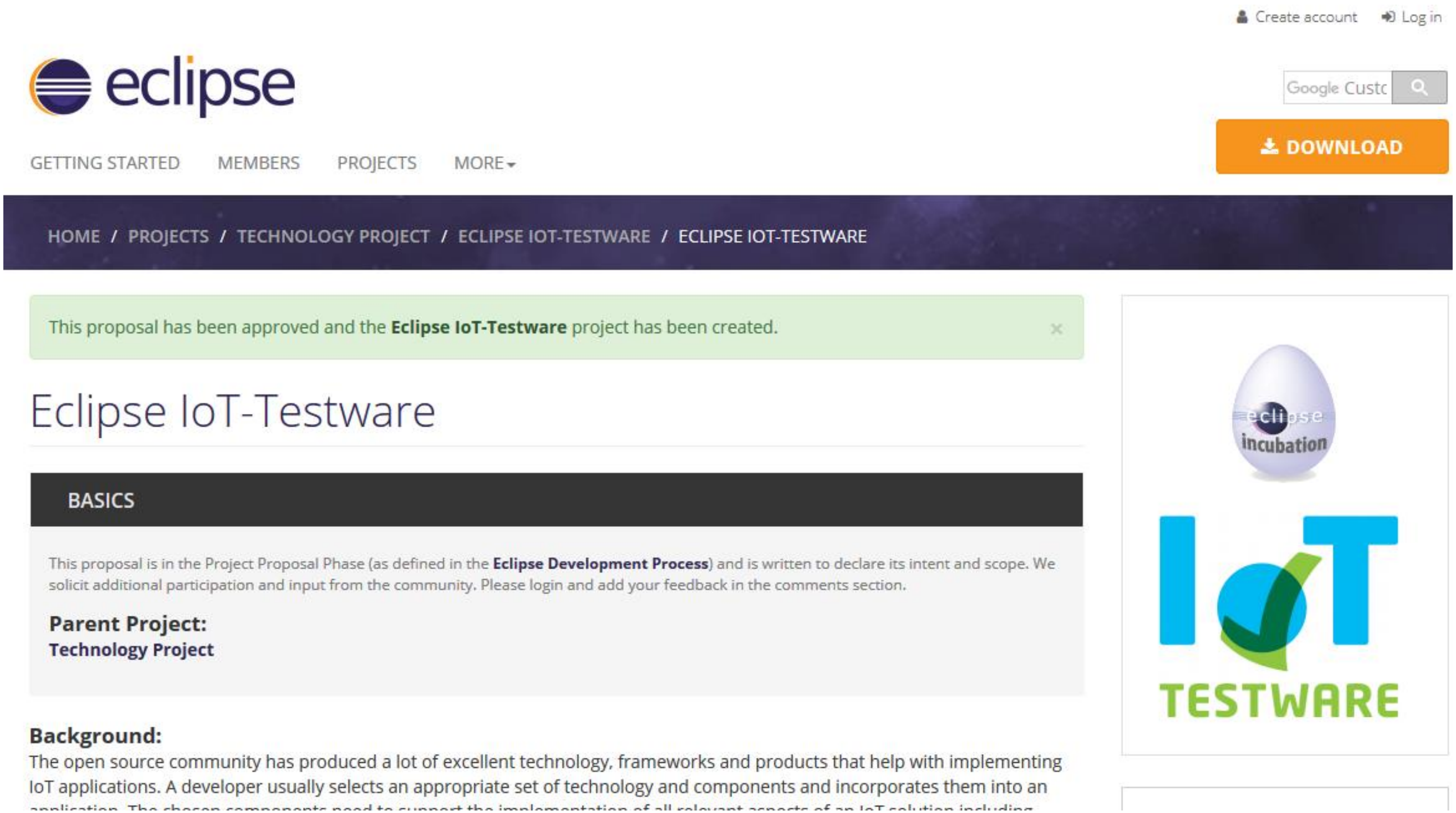
Ina Schieferdecker, Sascha Kretzschmann, Michael Wagner, Axel Rennoch

QRS, Praha, Czech Republic, July 27, 2017

IoT TESTWARE

Fraunhofer
FOKUS

THE ECLIPSE PROJECT



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

Eclipse IoT-Testware

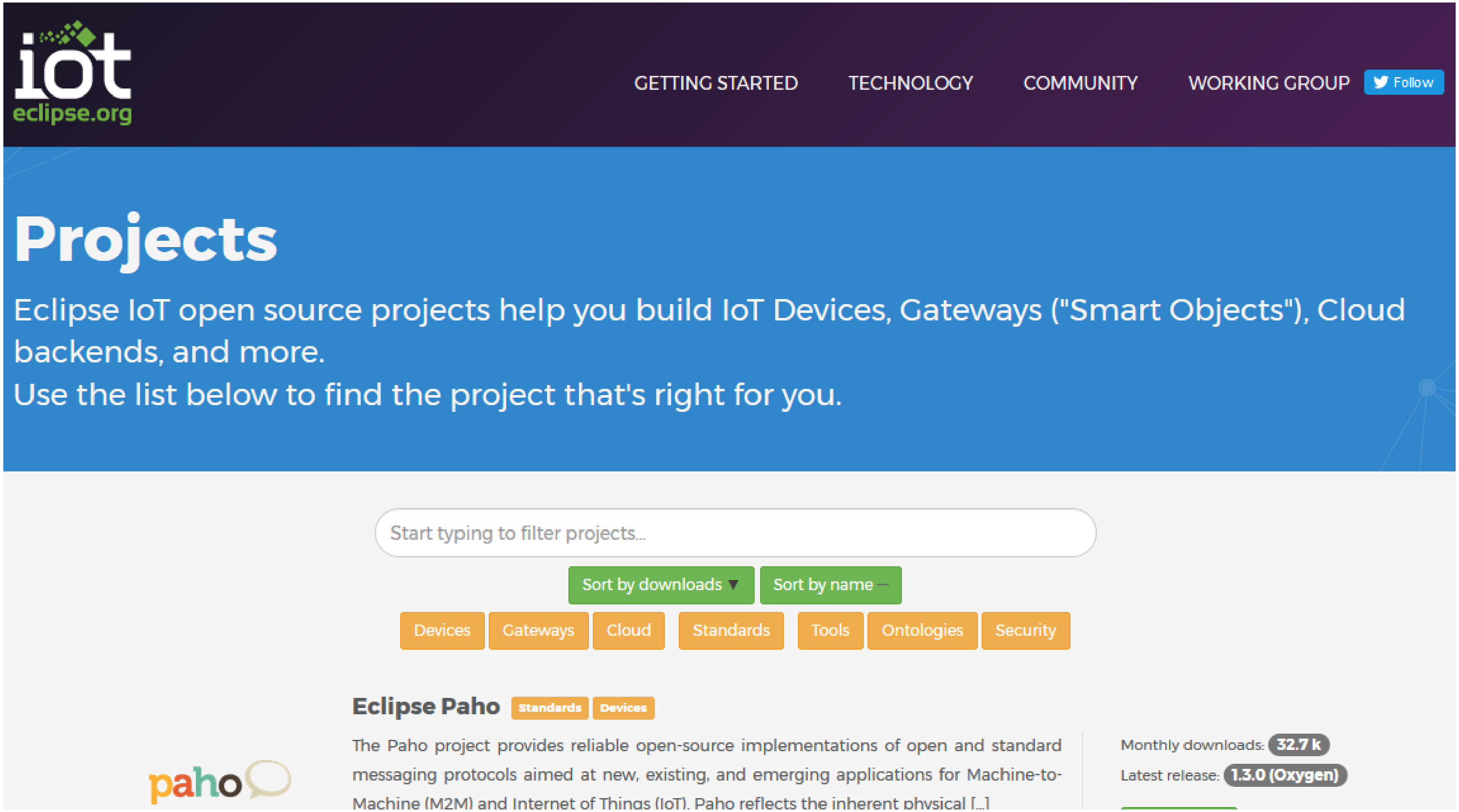
BASICS

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



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.

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

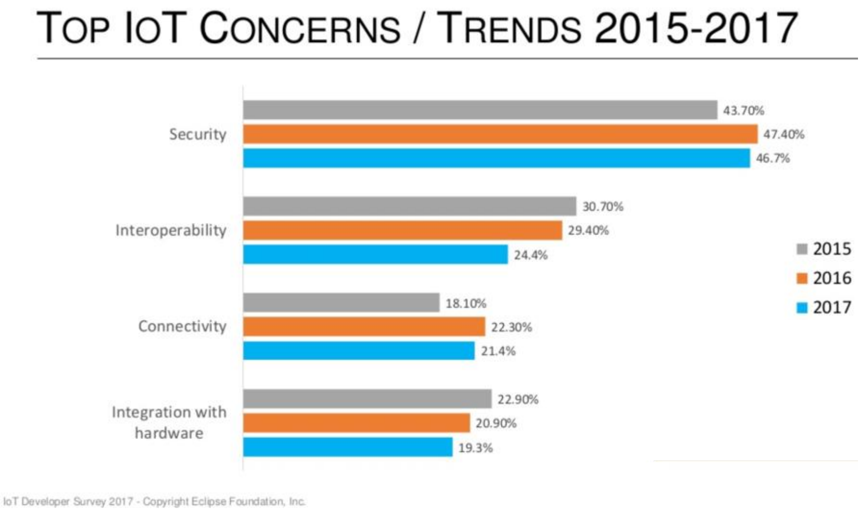
4. FOKUS contribution to IoT testing

5. Outlook

INTRODUCTION

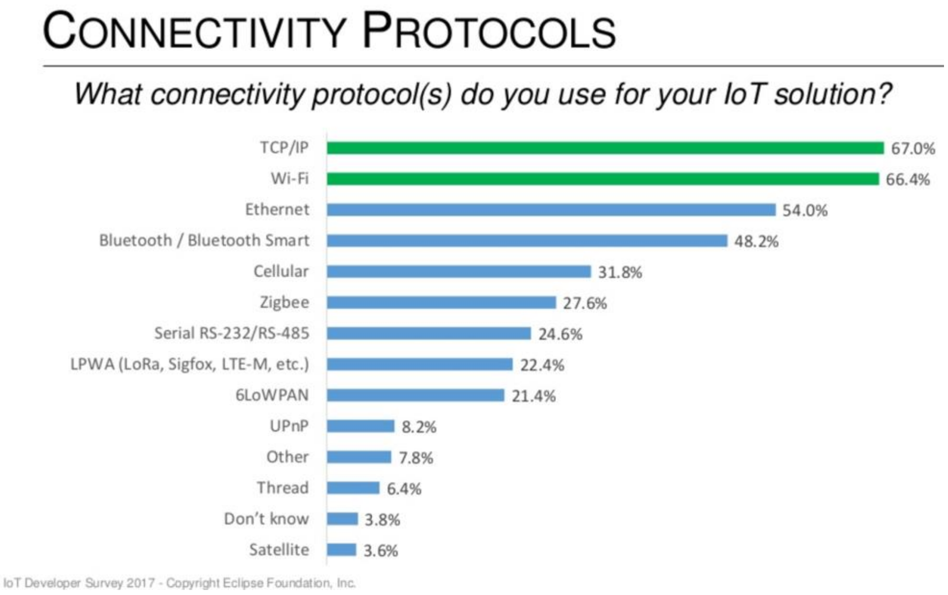
Where are we?

TRENDS IN IOT



Top IoT Concerns / Trends 2015-2017

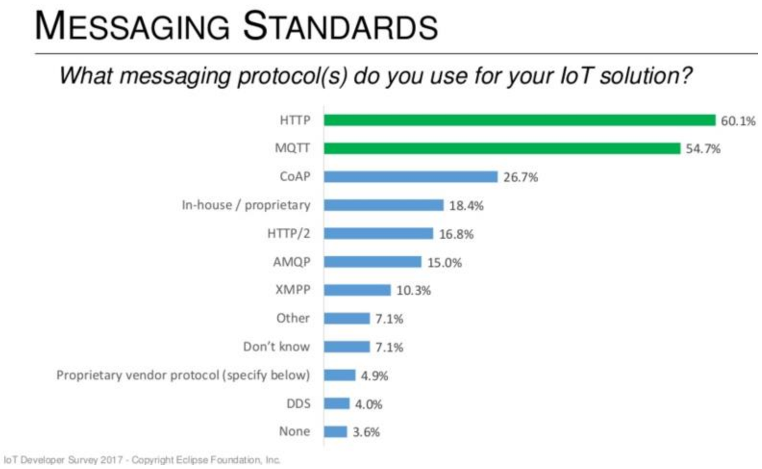
IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc.



Connectivity Protocols

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

IoT Developer Survey 2017 - Copyright Eclipse Foundation, Inc.

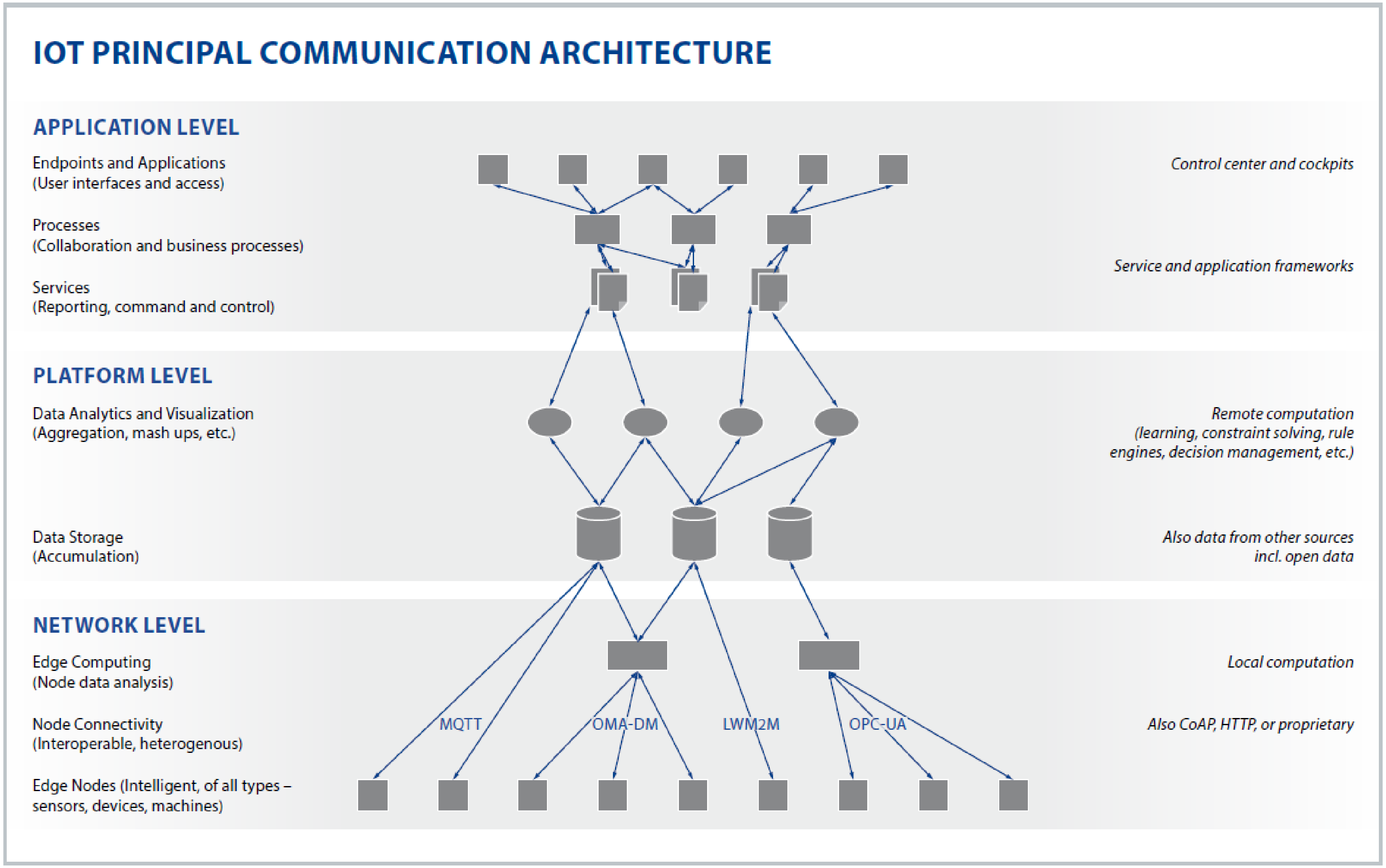


Messaging Standards

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)

Control center and cockpits

Processes
(Collaboration and business processes)

Service and application frameworks

Services
(Reporting, command and control)

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)

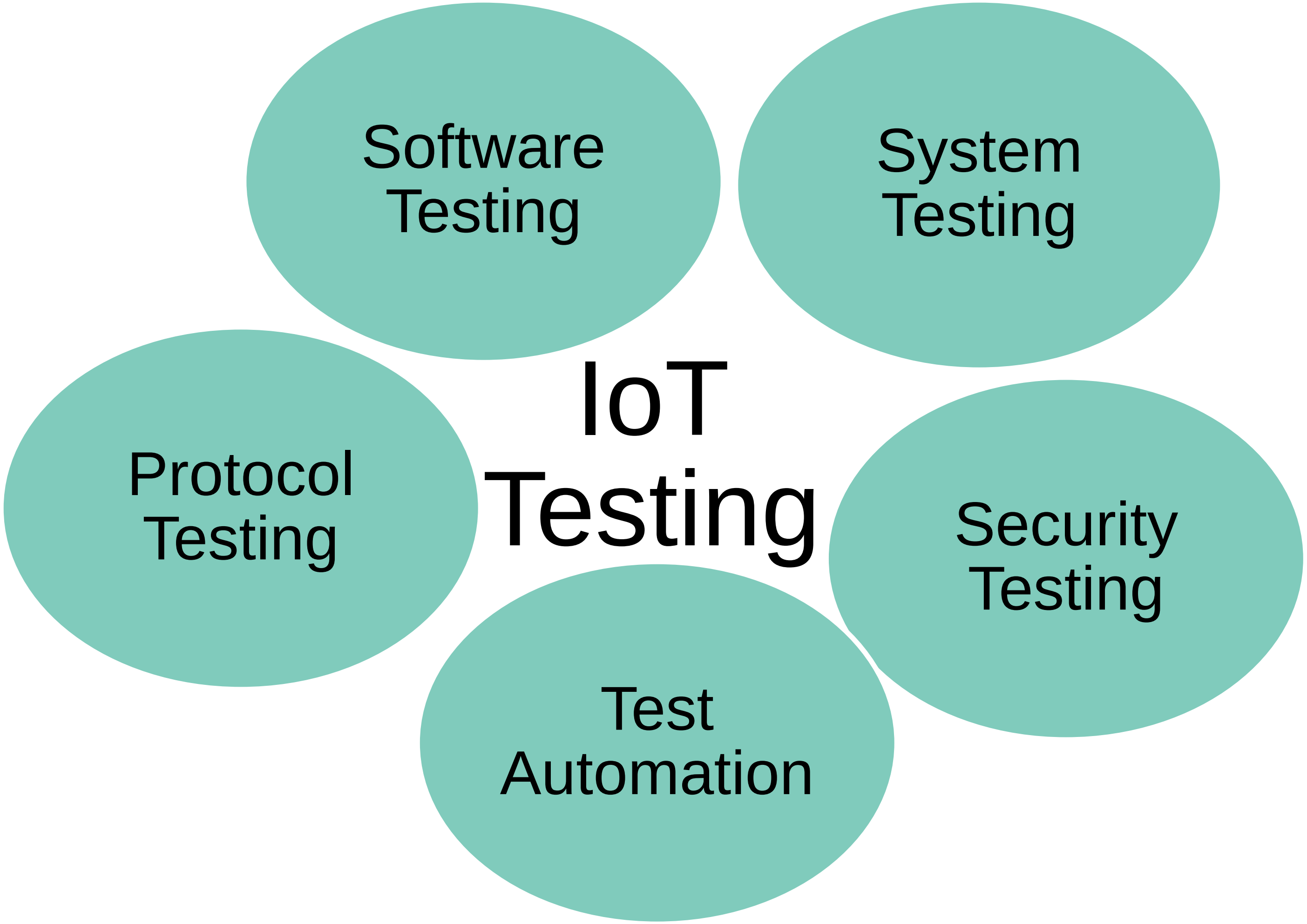
Local computation

Node Connectivity
(Interoperable, heterogenous)

Also CoAP, HTTP, or proprietary

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

INTEGRATION OF SEVERAL TESTING APPROACHES



IoT Testing

Software Testing

System Testing

Protocol Testing

Security Testing

Test Automation

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 r**esource-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?

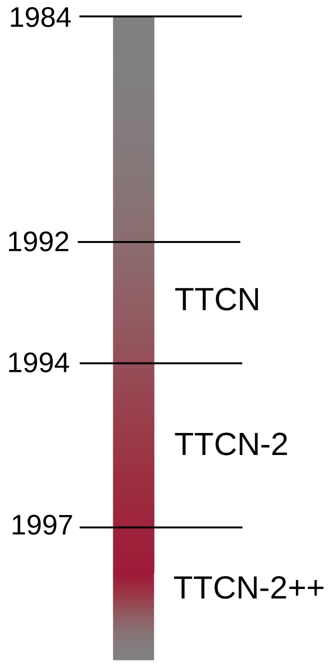
Fraunhofer
FOKUS

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

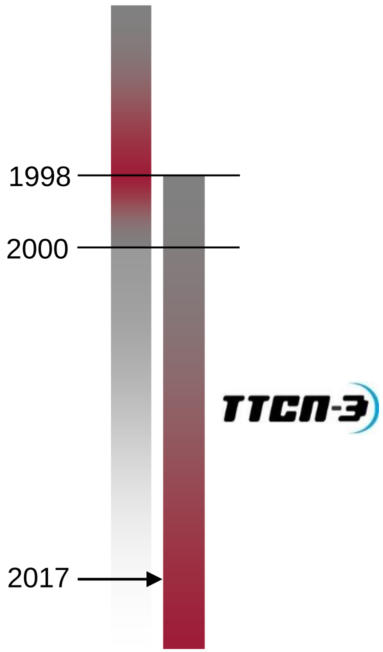


- TTCN (1992)
- published as ISO standard
- “Tree and Tabular Combined Notation”
- used for protocol tests:
  GSM, N-ISDN, B-ISDN

- TTCN-2/2++ (1997)
- enhancements by ETSI MTS
- module concept, concurrency
- used for conformance tests

Fraunhofer
FOKUS

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**

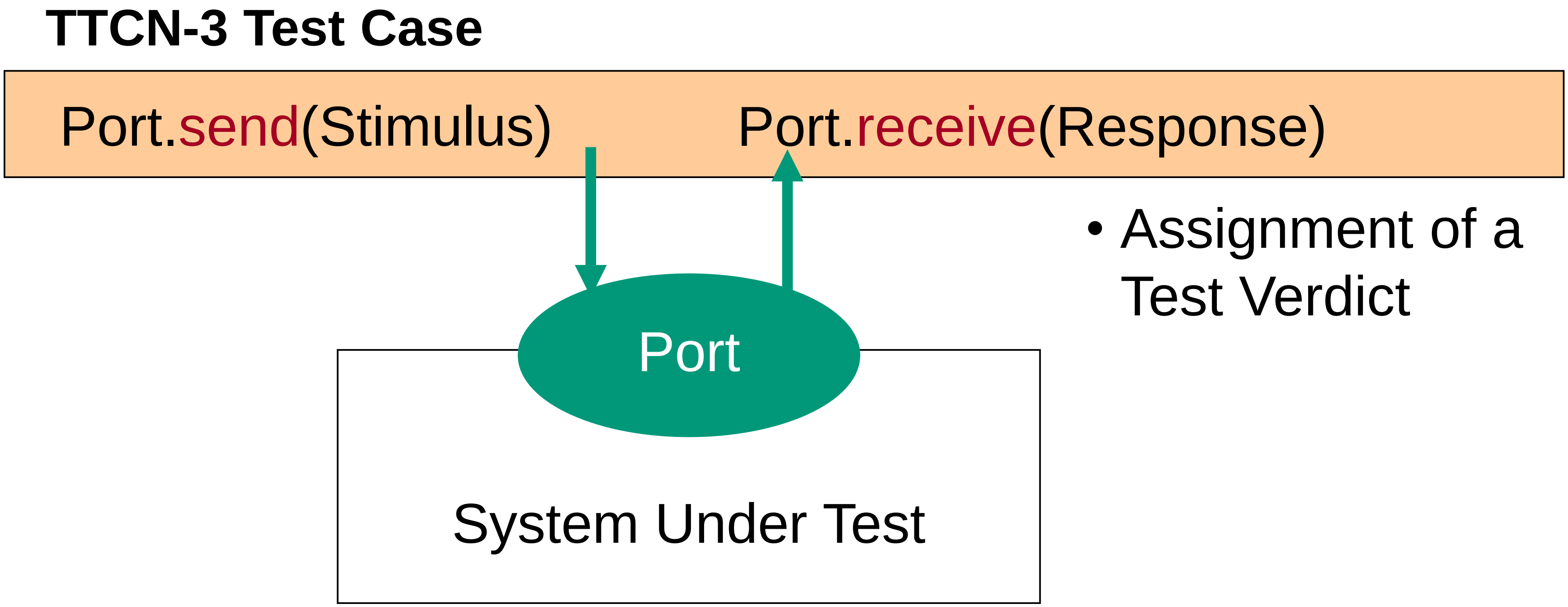
DESIGN PRINCIPLES OF TTCN-3

- **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



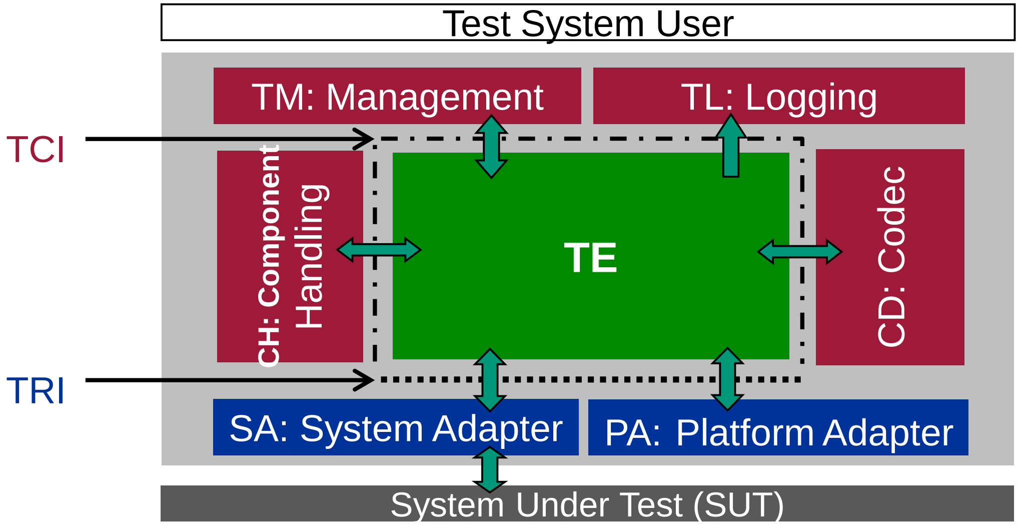
- Assignment of a Test Verdict

MAJOR LANGUAGE ELEMENTS OF TTCN-3 NOTATION

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/expected during test execution (templates, values)
Test Configuration	Definition of the test components and **communication ports**
Test Behavior	Specification of the **dynamic** test behavior

A TTCN-3 TEST SYSTEM

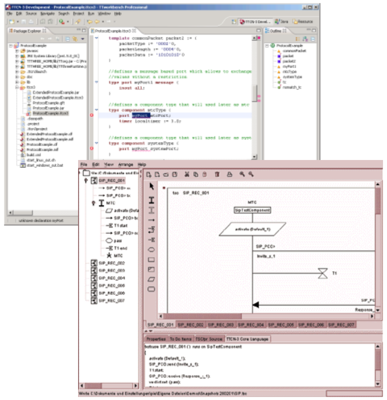


TE	–	TTCN-3 Executable
TM	–	Test Management
TL	–	Test Logging
CD	–	Codec
CH	–	Component Handling
SA	–	System Adapter
PA	–	Platform Adapter
SUT	–	System Under Test

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

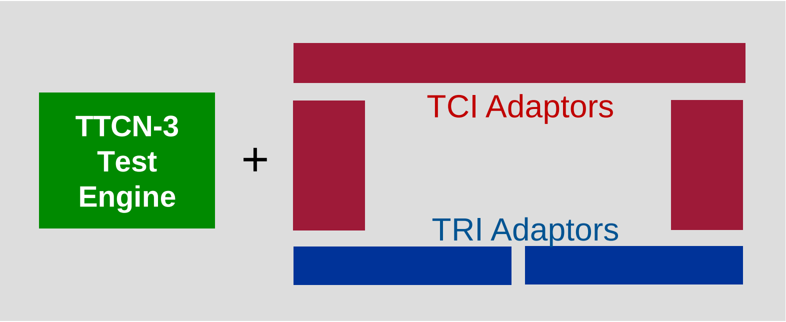


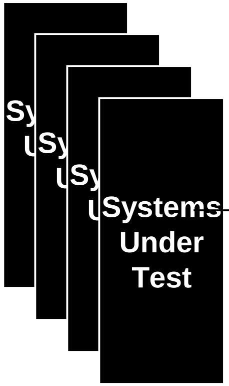
```
template commonPacket packet2 := {
    packetType := '0002'O,
    packetLength := '0004'O,
    packetData := '10101010'O
}

//defines a message based port which allows to exchange
//values without a restriction
type port myPort1 message {
    inout all;
}

//defines a component type that will used later as mtc
type component mtcType {
    port myPort mtcPort;
    timer localtimer := 3.0;
}

//defines a component type that will used later as syst
type component systemType {
    port myPort systemPort;
}
```

```
testcase SIP_REC_001 () runs on SipTestComponent
{
  activate (Default_1);
  SIP_PCO.send (Invite_s_1);
  T1.start;
  SIP_PCO.receive (Response_r_1);
  verdict.set (pass);
```

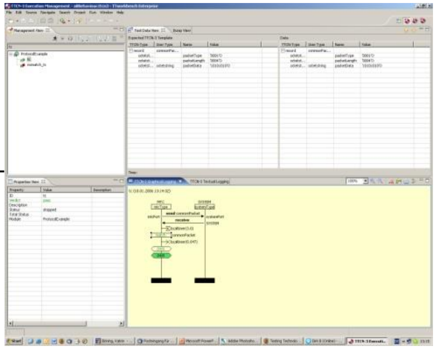
TTCN-3

Test System



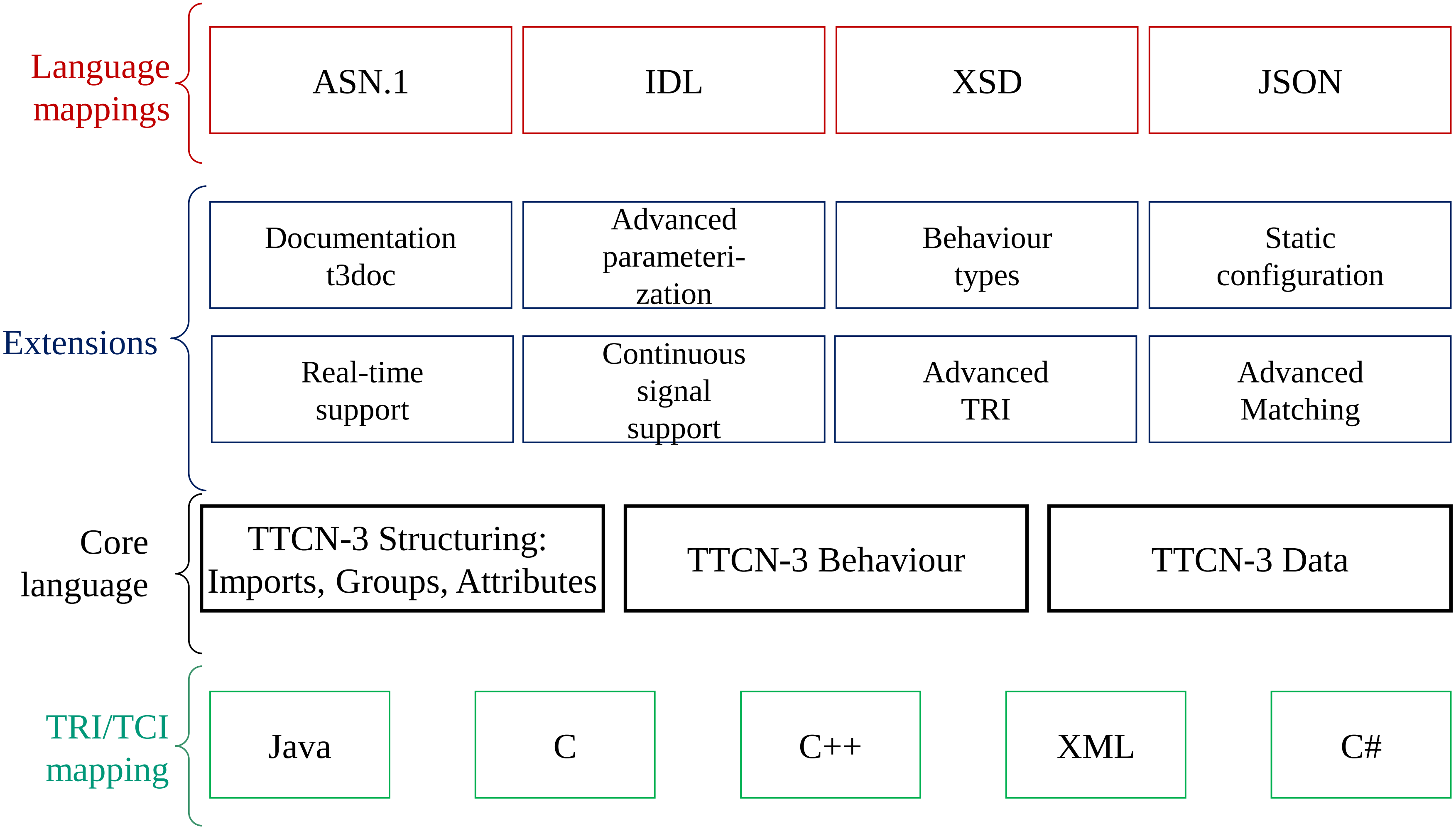


Communication / Invocation



Automated Test Execution and Reporting

TTCN-3 TECHNOLOGY OVERVIEW



Language mappings

ASN.1

IDL

XSD

JSON

Extensions

Documentation t3doc

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++

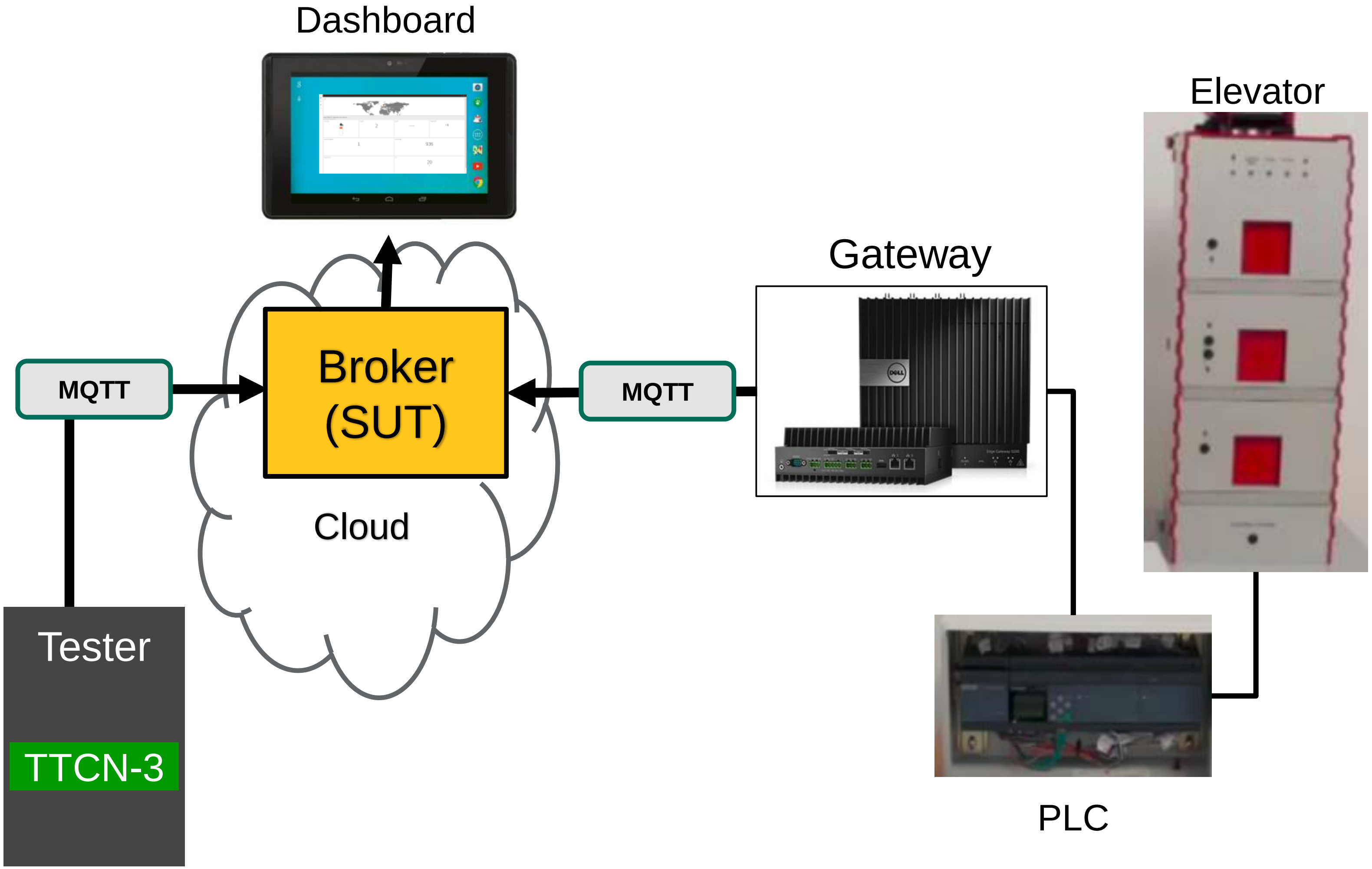
XML

C#

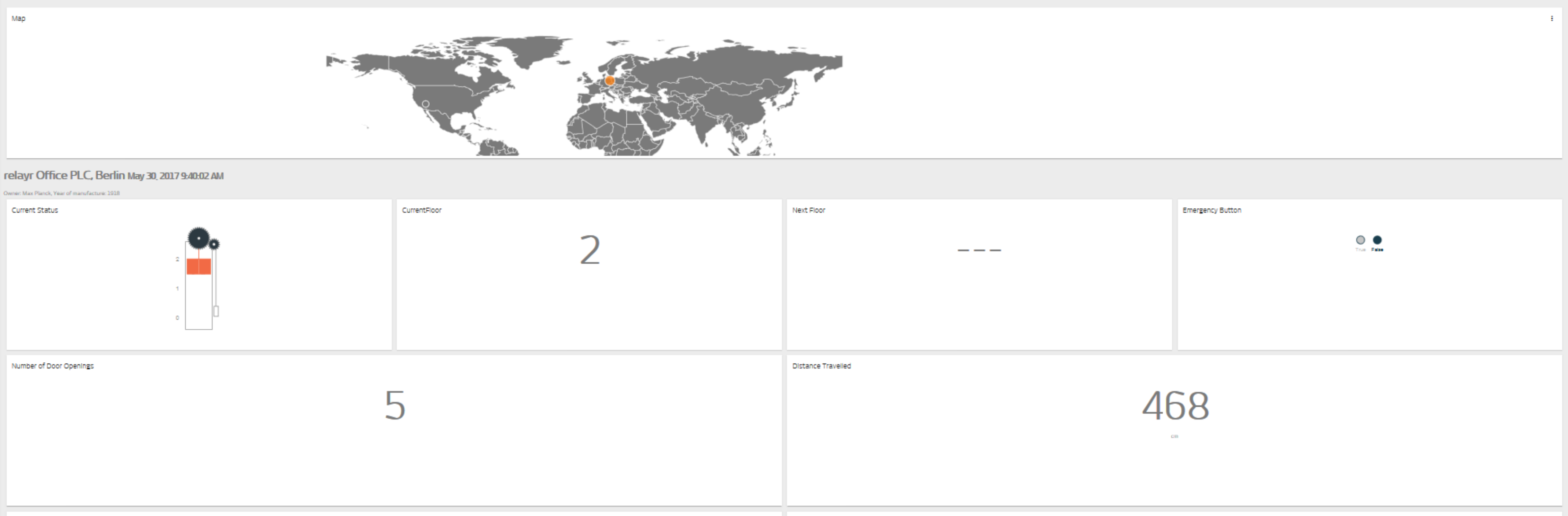
TTCN-3 IN USE

How do we use it?

ELEVATOR DEMO CONFIGURATION



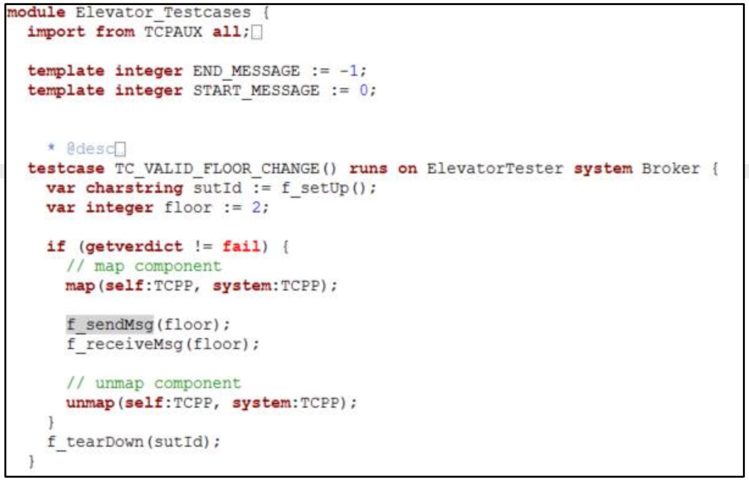
ELEVATOR DEMO CONFIGURATION (CONT.)



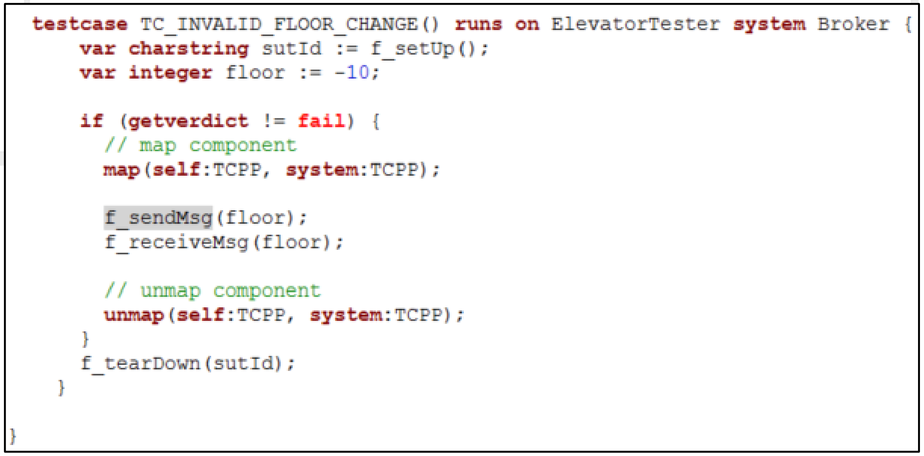
Map

relayr Office PLC, Berlin May 30, 2017 9:40:02 AM

Owner: Max Planck, Year of manufacture: 1918



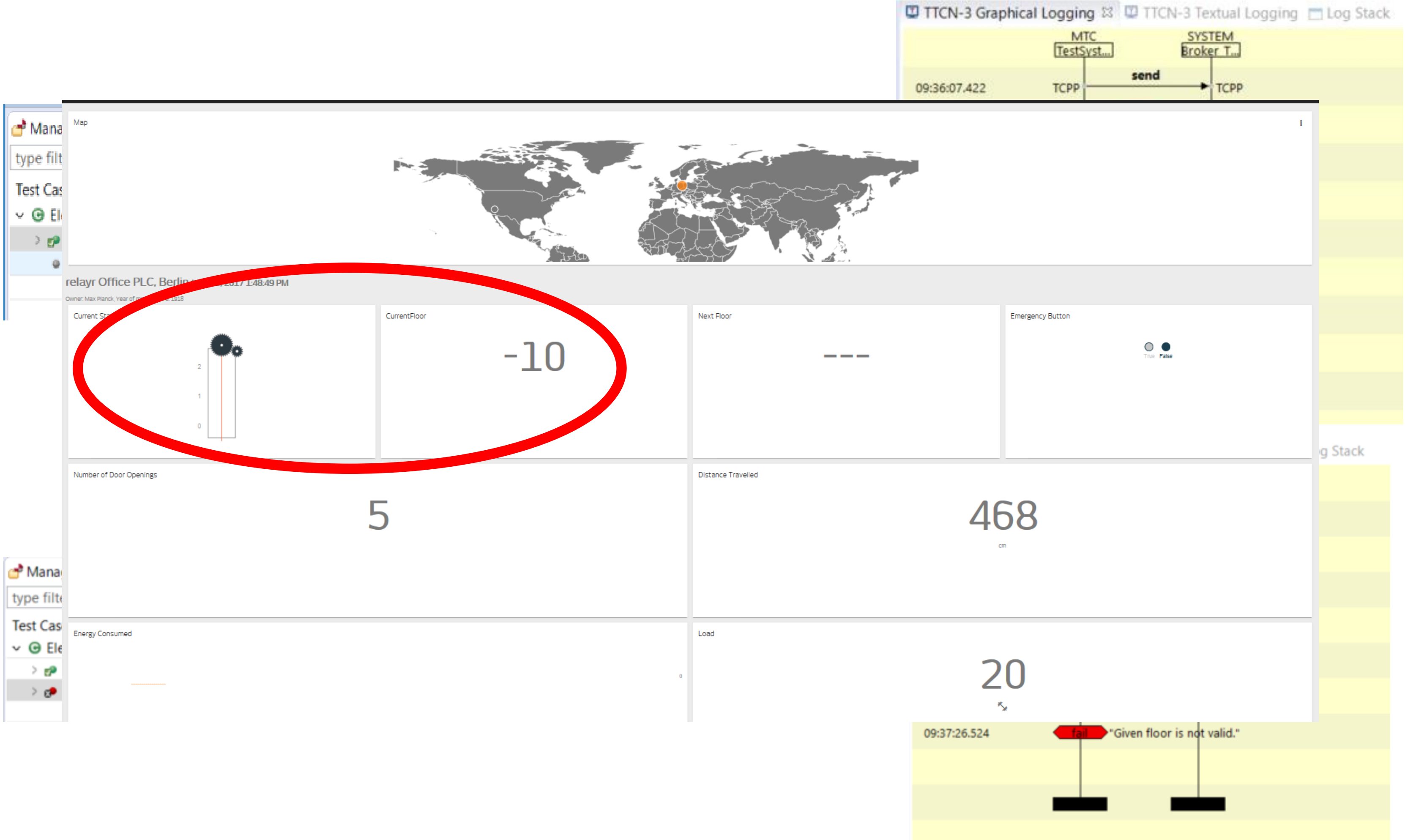
```
module Elevator_Testcases {
  import from TCPAUX all;

  template integer END_MESSAGE := -1;
  template integer START_MESSAGE := 0;


   * @desc
  testcase TC_VALID_FLOOR_CHANGE() runs on ElevatorTester system Broker {
    var charstring sutId := f_setUp();
    var integer floor := 2;

    if (getverdict != fail) {
      // map component
      map(self:TCPP, system:TCPP);

      f_sendMsg(floor);
      f_receiveMsg(floor);

      // unmap component
      unmap(self:TCPP, system:TCPP);
    }
    f_tearDown(sutId);
  }
```



```
  testcase TC_INVALID_FLOOR_CHANGE() runs on ElevatorTester system Broker {
      var charstring sutId := f_setUp();
      var integer floor := -10;

      if (getverdict != fail) {
        // map component
        map(self:TCPP, system:TCPP);

        f_sendMsg(floor);
        f_receiveMsg(floor);

        // unmap component
        unmap(self:TCPP, system:TCPP);
      }
      f_tearDown(sutId);
    }

}
```

Fraunhofer
FOKUS

ELEVATOR DEMO CONFIGURATION (CONT.)



09:36:07.422

relayr Office PLC, Berlin ... 2017 1:48:49 PM

Owner: Max Planck, Year of ... 1918

CurrentFloor: -10

Next Floor: ---

Number of Door Openings: 5

Distance Travelled: 468 cm

Load: 20

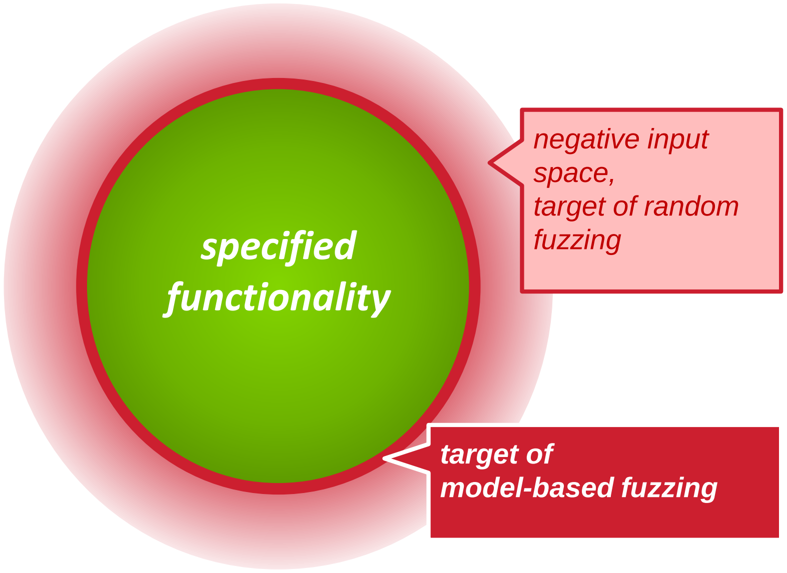
09:37:26.524 fail "Given floor is not valid."

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



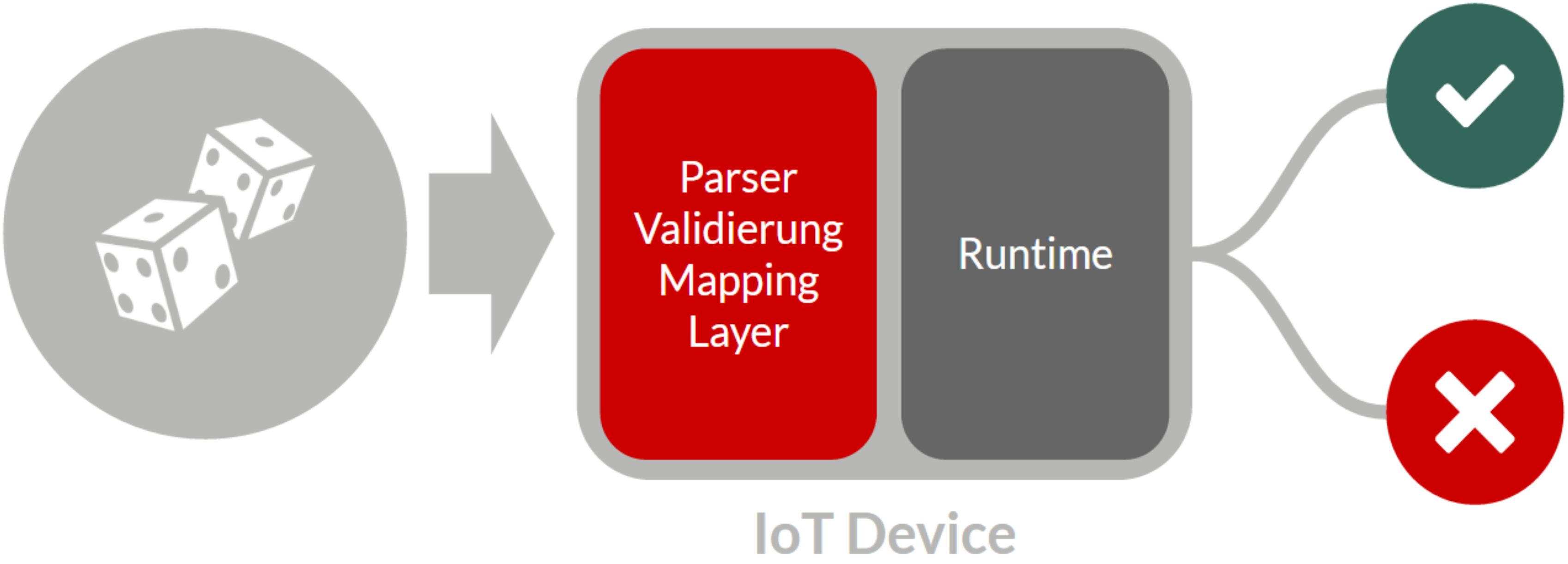
specified functionality

negative input space, target of random fuzzing

target of model-based fuzzing

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

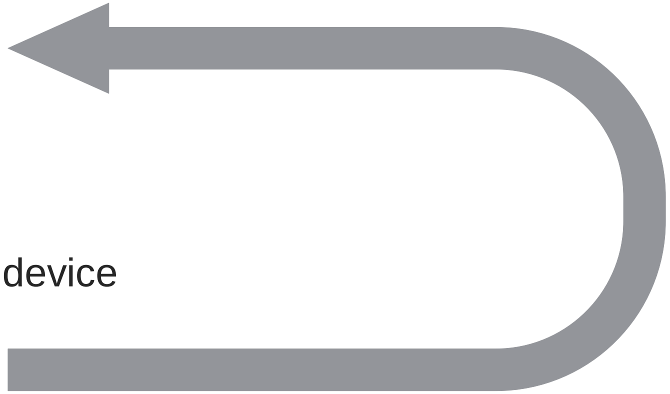
FUZZINO

https://github.com/fraunhoferfokus/Fuzzino

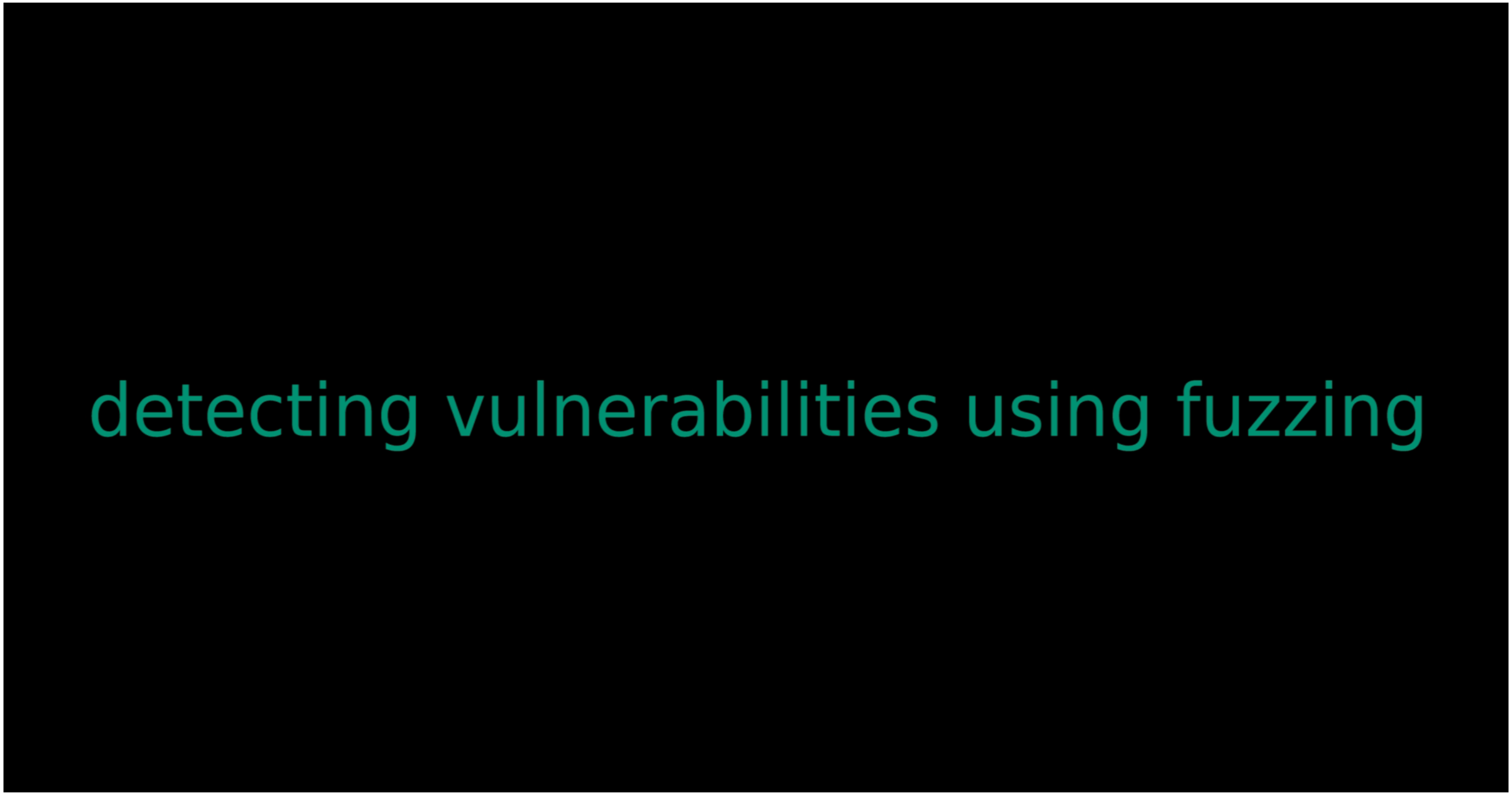
- Supports generation and mutation based fuzzing
- Platform-independent: is implemented in Java
- Language-independent: provides an XML-based interface
- Automated: automatically selects appropriate fuzzing heuristics
- Efficient & scalable: the user can decide which fuzzing heuristics shall be used
- 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)



detecting vulnerabilities using fuzzing

FOKUS CONTRIBUTION TO IOT TESTING

What else?

Fraunhofer
FOKUS

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

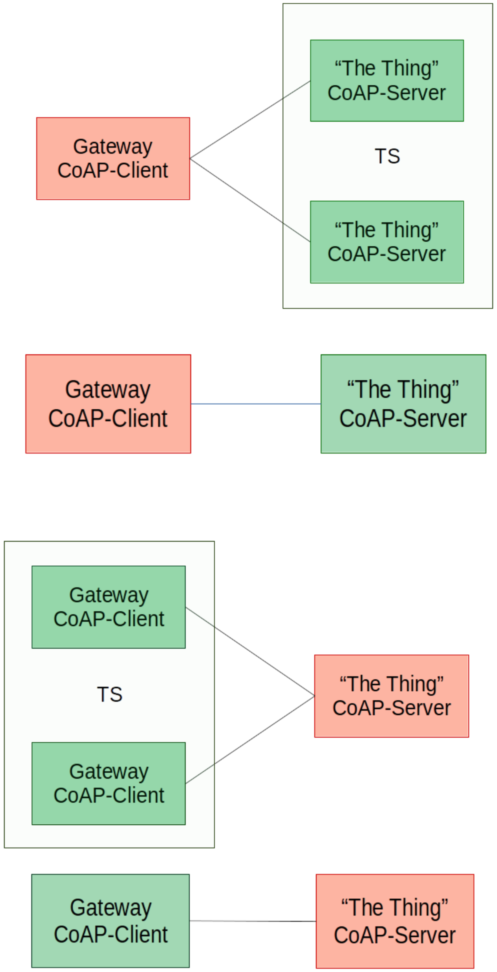
- Approved by Eclipse Foundation :
  https://projects.eclipse.org/proposals/eclipse-iot-testware

- Creation of TTCN-3 test suites for CoAP and MQTT
- Project partners: relayr GmbH, Ericsson, LAAS/CNRS, itemis AG, Spirent Communications, Easy Global Market

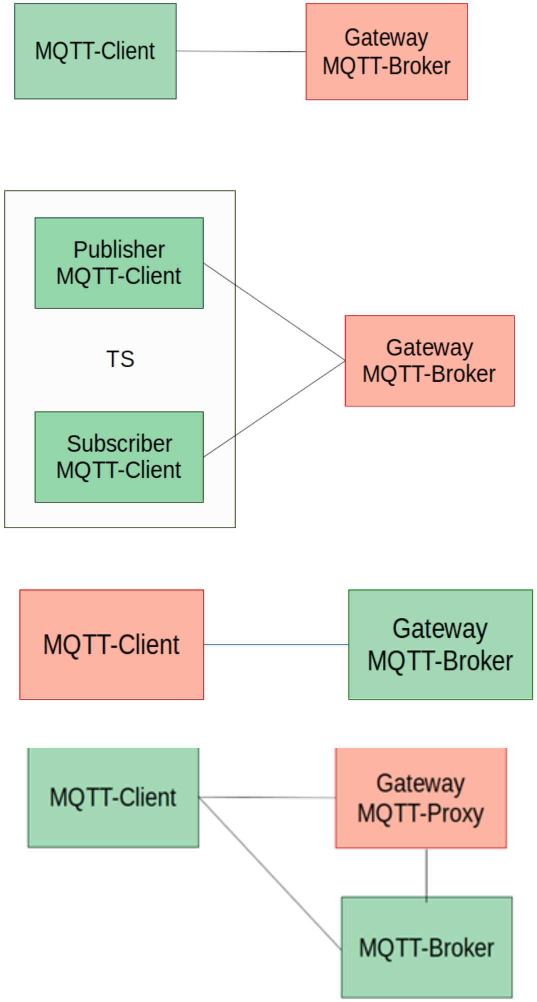
- Current schedule
  - 2017Q2: creation of a catalogue for test objectives (test purposes)
  - 2017Q3: initial publication of implemented TTCN-3 tests

TEST CONFIGURATIONS

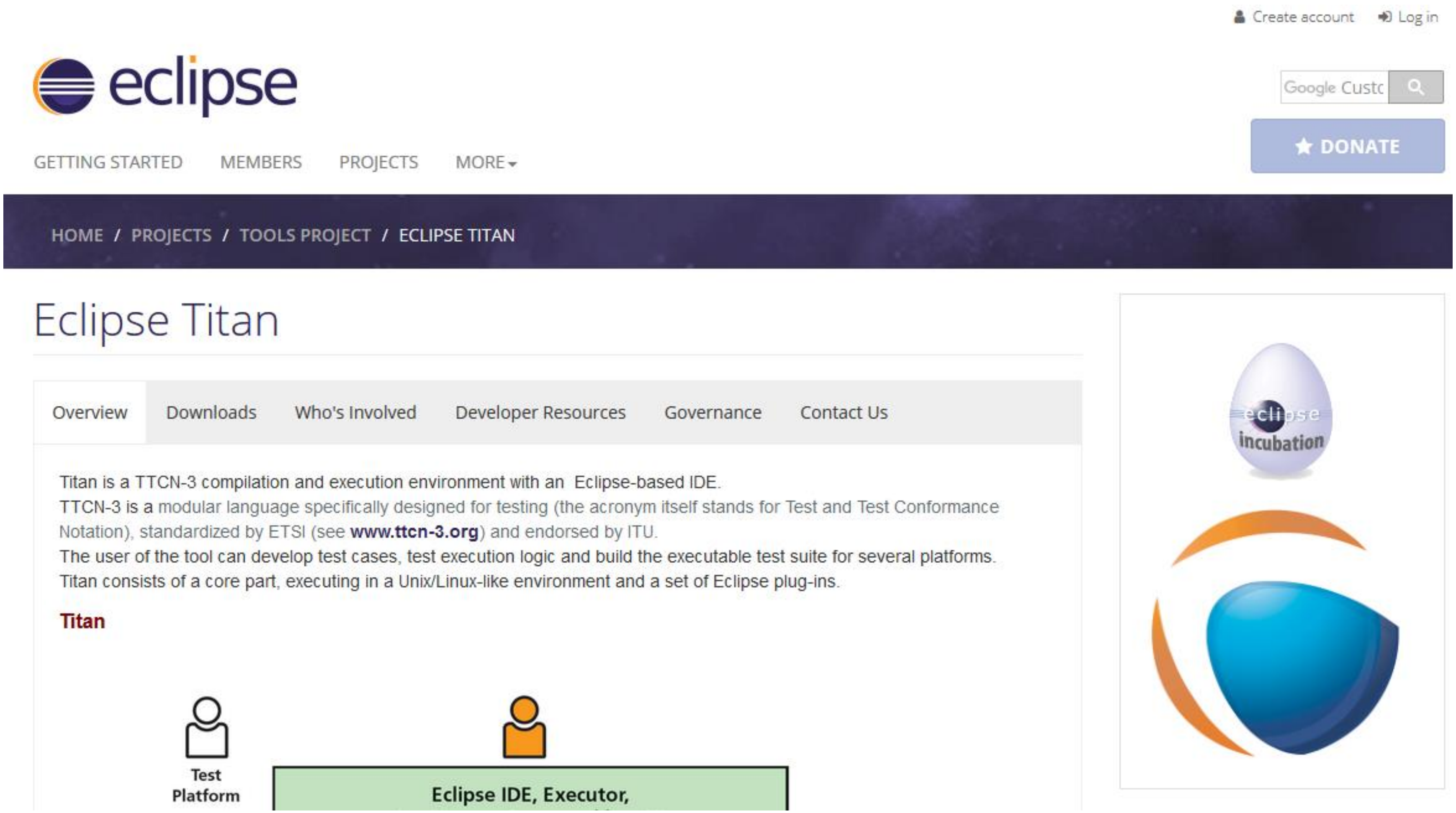
CoAP



MQTT



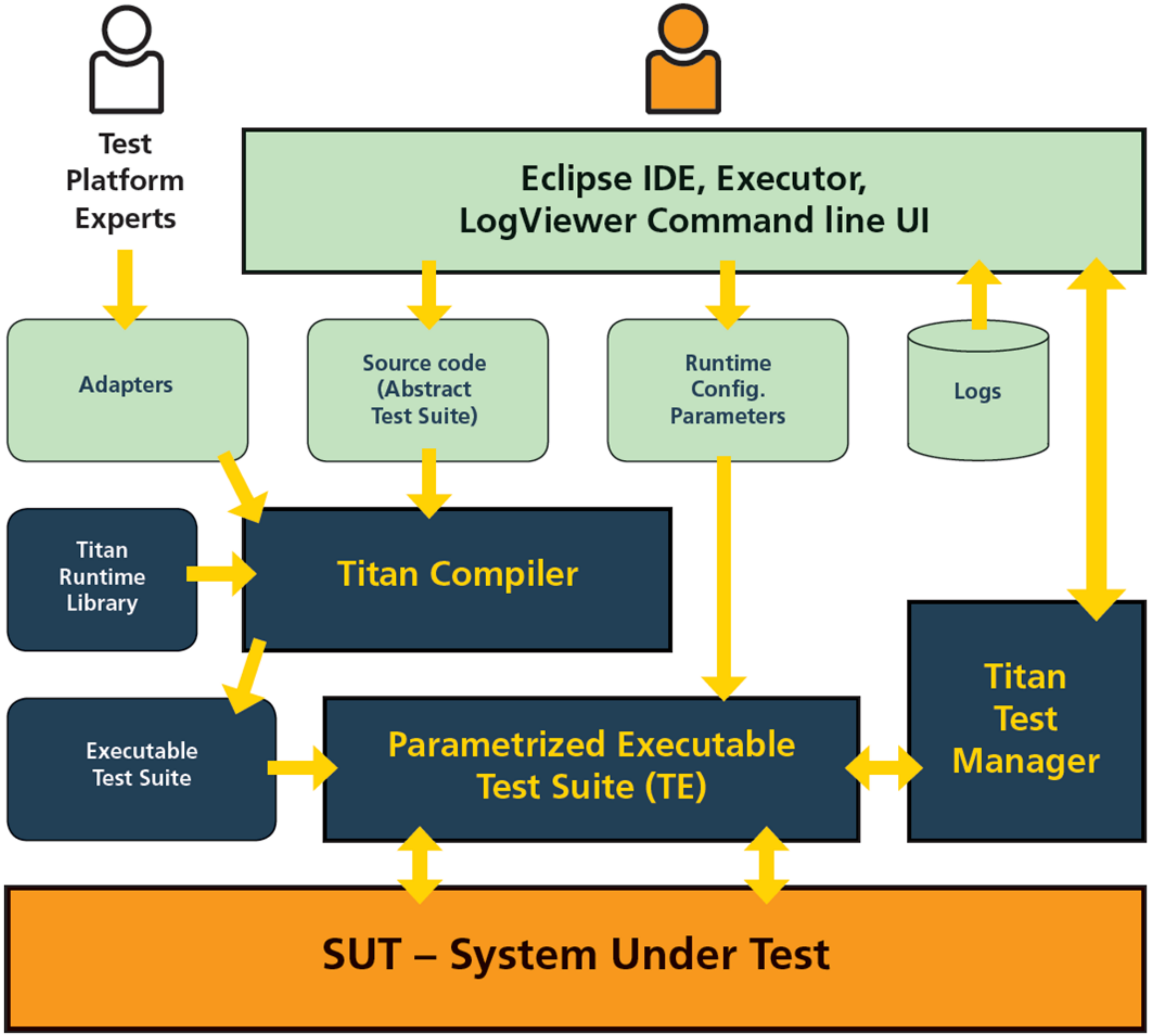
THE TEST EXECUTION TOOL



Eclipse Titan

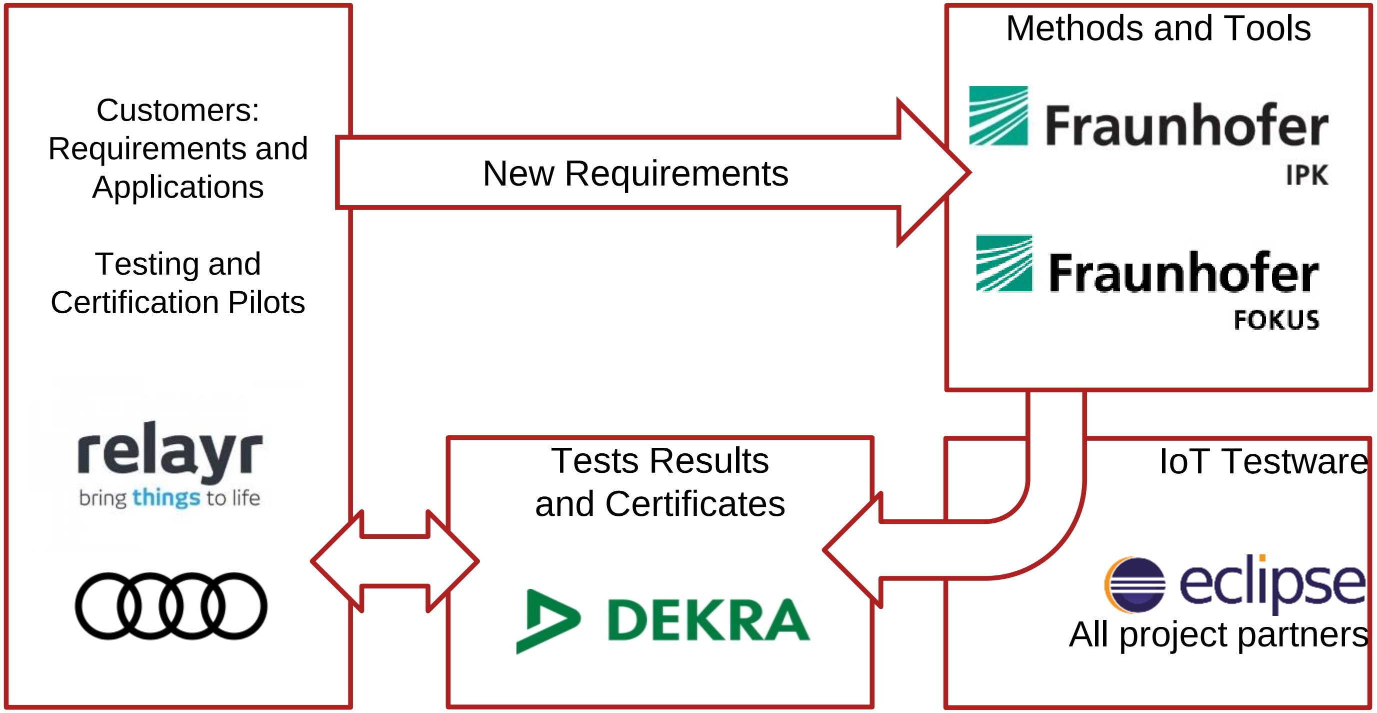
Titan is a TTCN-3 compilation and execution environment with an Eclipse-based IDE.
TTCN-3 is a modular language specifically designed for testing (the acronym itself stands for Test and Test Conformance Notation), standardized by ETSI (see **www.ttcn-3.org**) and endorsed by ITU.
The user of the tool 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


Test Platform Experts
Eclipse IDE, Executor, LogViewer Command line UI
Adapters
Source code (Abstract Test Suite)
Runtime Config. Parameters
Logs
Titan Runtime Library
Titan Compiler
Executable Test Suite
Parametrized Executable Test Suite (TE)
Titan Test Manager
SUT – System Under Test

IOT-T PROJECT

Federal Ministry
for Economic Affairs
and Energy



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

Fraunhofer FOKUS

OUTLOOK

What are further ideas?

Fraunhofer
FOKUS

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

TTCN-3 → TTCN-3
virtualized

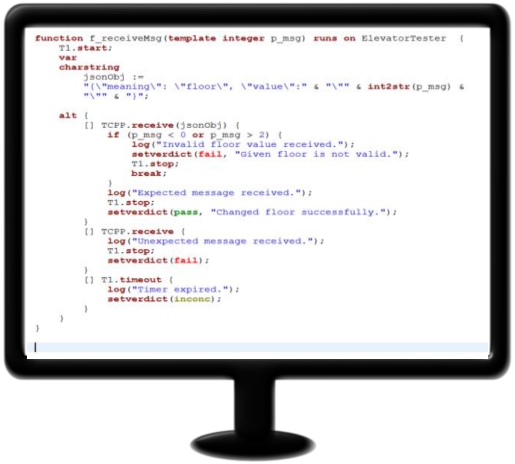
- Easy solution to write your test cases “online“
- Deploy your test suite (Java, C++ or as service)
- Run the executables

+ Hide complexity → everyone can write tests

+ Test implementation is straight forward

- Tests may not running on highly constrained devices

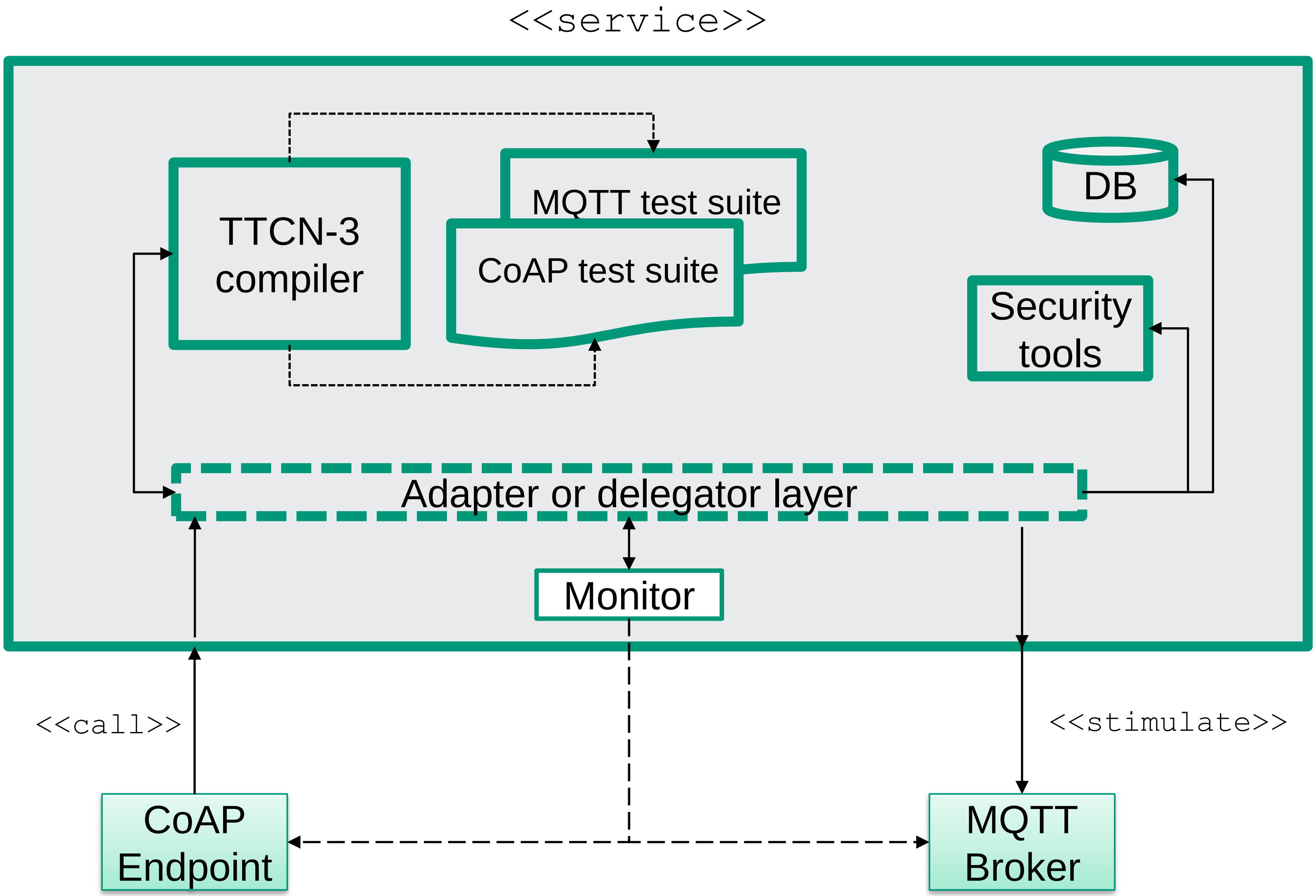
- Still difficult to configure other parts of the test system



```
function f_receiveMsg(template integer p_msg) runs on ElevatorTester  {
    T1.start;
    var
    charstring
        jsonObj :=
        "{\"meaning\": \"floor\", \"value\":" & "\"" & int2str(p_msg) &
        "\"" & "}";

    alt {
        [] TCPP.receive(jsonObj) {
            if (p_msg < 0 or p_msg > 2) {
                log("Invalid floor value received.");
                setverdict(fail, "Given floor is not valid.");
                T1.stop;
                break;
            }
            log("Expected message received.");
            T1.stop;
            setverdict(pass, "Changed floor successfully.");
        }
        [] TCPP.receive {
            log("Unexpected message received.");
            T1.stop;
            setverdict(fail);
        }
        [] T1.timeout {
            log("Timer expired.");
            setverdict(inconc);
        }
    }
}
```

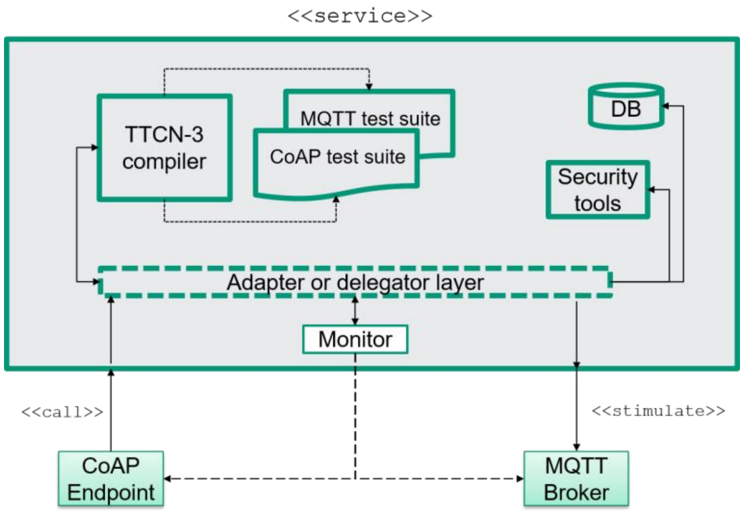
Fraunhofer
FOKUS

VIRTUALIZED TESTING



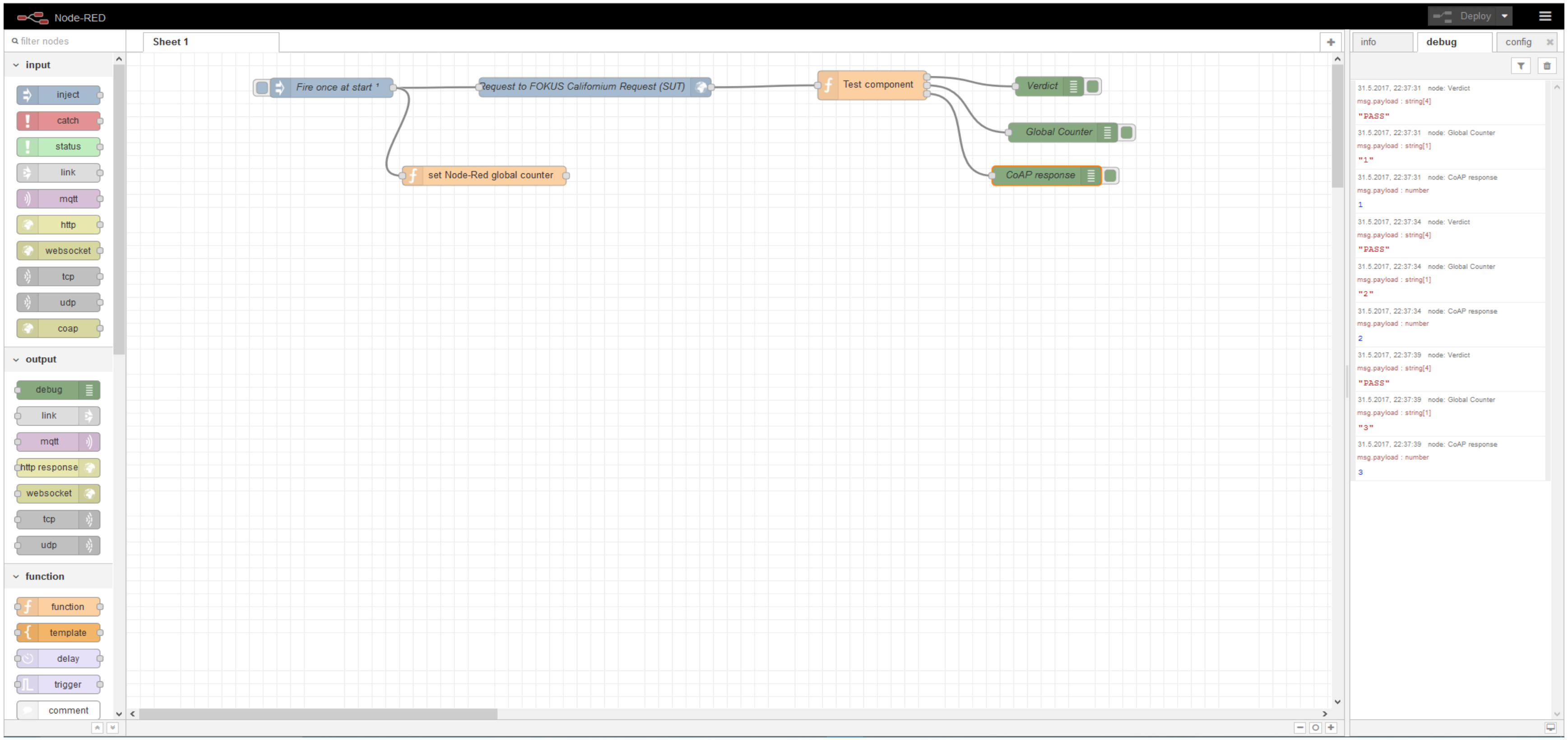
PROS AND CONS OF VIRTUALIZED TESTING

- + 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



```
31.5.2017, 22:37:31   node: Verdict
msg.payload : string[4]
"PASS"
31.5.2017, 22:37:31   node: Global Counter
msg.payload : string[1]
"1"
31.5.2017, 22:37:31   node: CoAP response
msg.payload : number
1
31.5.2017, 22:37:34   node: Verdict
msg.payload : string[4]
"PASS"
31.5.2017, 22:37:34   node: Global Counter
msg.payload : string[1]
"2"
31.5.2017, 22:37:34   node: CoAP response
msg.payload : number
2
31.5.2017, 22:37:39   node: Verdict
msg.payload : string[4]
"PASS"
31.5.2017, 22:37:39   node: Global Counter
msg.payload : string[1]
"3"
31.5.2017, 22:37:39   node: CoAP response
msg.payload : number
3
```

Fraunhofer
FOKUS

Thank you
for your attention!

www.fokus.fraunhofer.de

(System Quality Center)

CONTACTS

Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31
10589 Berlin, Germany

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