



Fundamentals of IoT Testing

Prof. Dr.-Ing. Ina Schieferdecker
Axel Rennoch

-
- Our Context
 - Introduction
 - Fundamentals (test objectives, test levels, etc.)
 - Eclipse IoT Testware

A photograph of the Fraunhofer FOKUS building at night. The building is a modern, multi-story structure with a curved corner and many windows that are illuminated from within, creating a warm glow against the dark blue sky. The building is surrounded by trees and a street with some blurred light trails from passing vehicles. The overall scene is a long-exposure shot, giving it a sense of motion and activity.

Fraunhofer FOKUS
Institute for Open Communication Systems

We connect everything
secure, reliable, trustworthy

We connect



Things



Systems



Processes

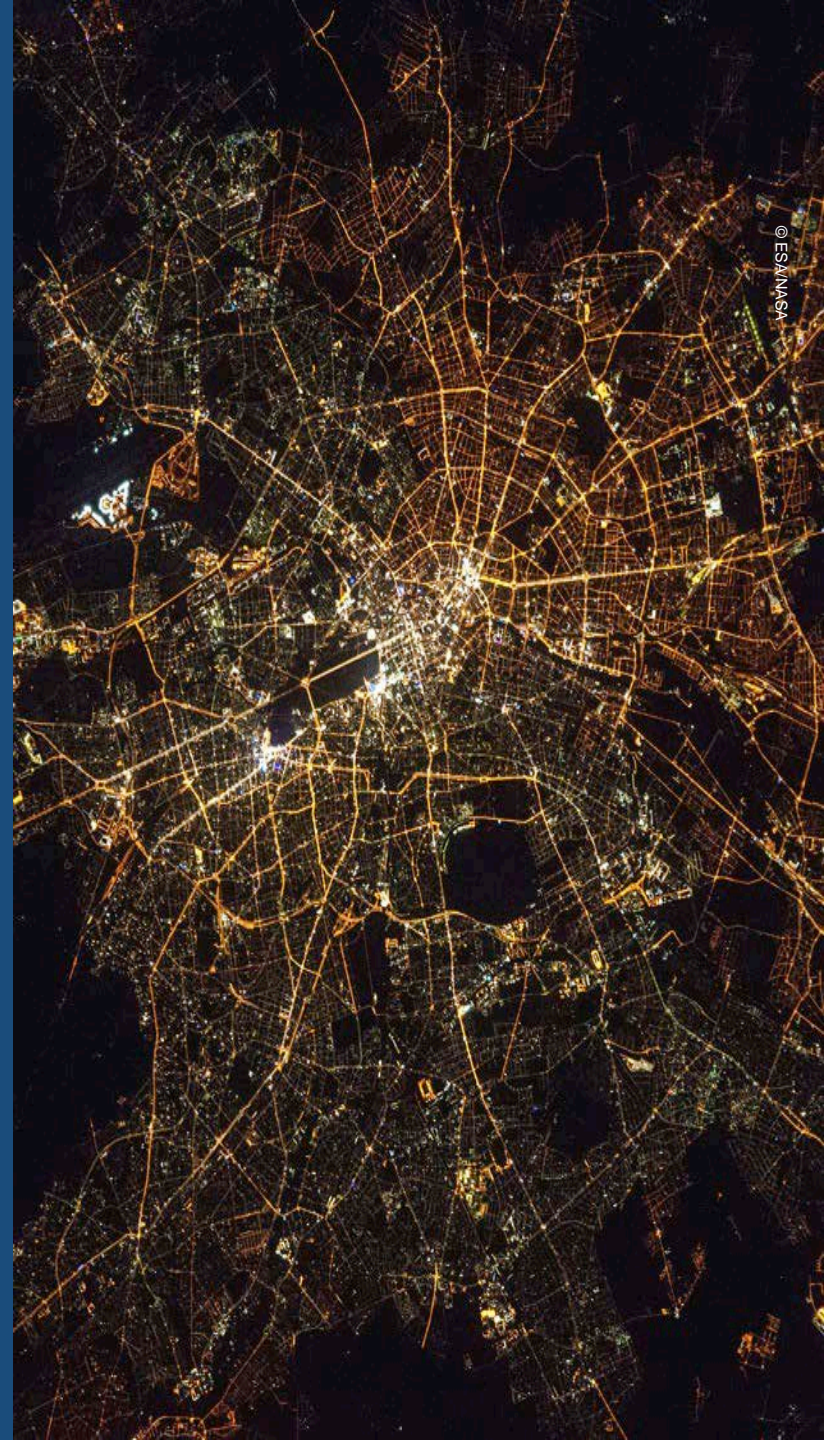


Persons



Organizations

- thematically
- technologically
- scientifically
- organizationally
- specialized
- secure
- effective

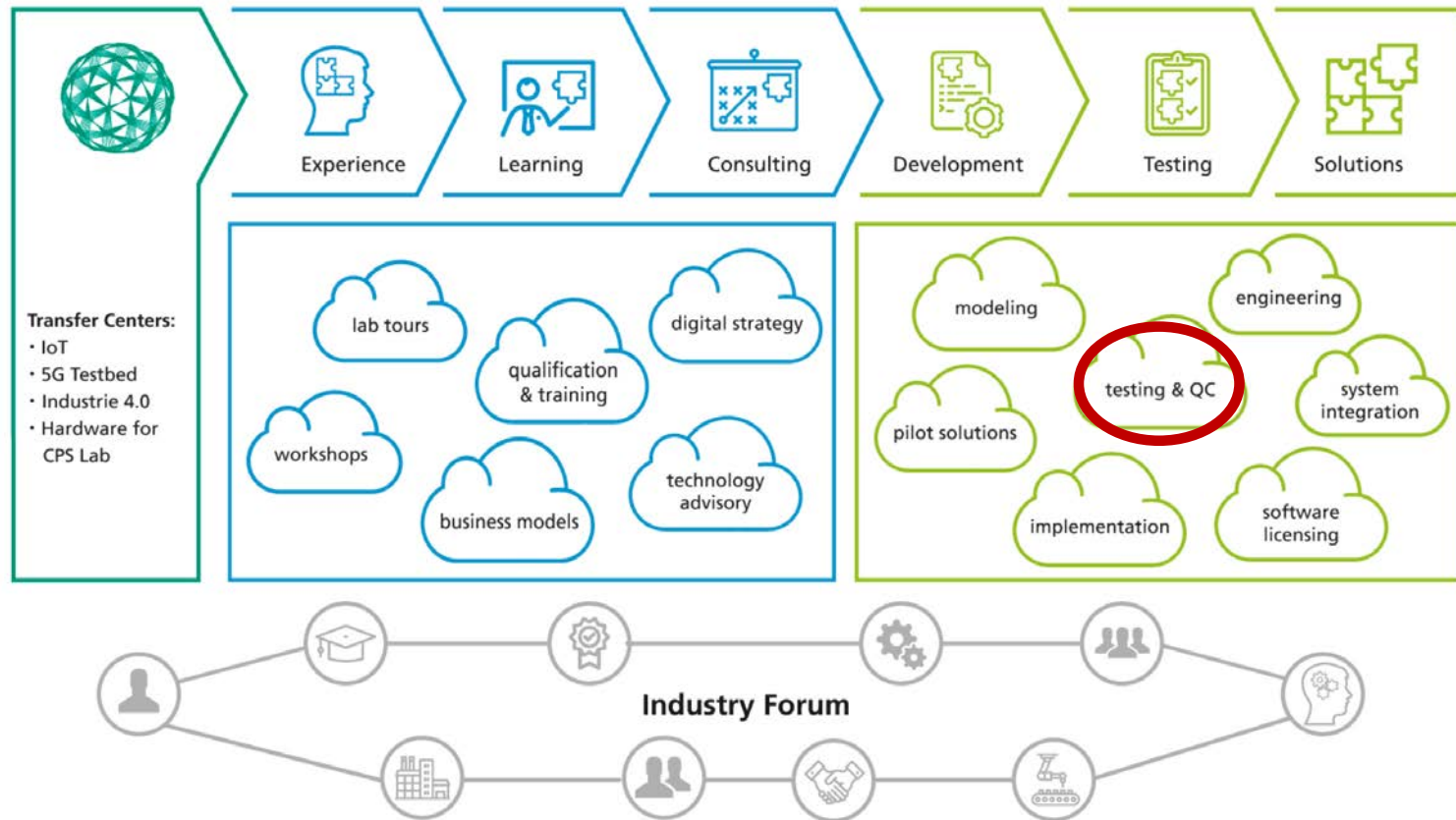


Cooperation



BERLIN CENTER FOR DIGITAL TRANSFORMATION

Digital Transformation from A to Z



International Software Testing Qualifications Board (www.istqb.org):

- Non-profit
- Founded in 2002
- Headquartered in Belgium
- Composed of volunteer international Testing Experts from almost 50 National Boards
- Responsible for the “ISTQB® Certified Tester” scheme worldwide
- Provides **software testing glossary**
- Already over 600.000 exams and 450.000 certificates worldwide

Advancing
the software testing profession



Fundamentals of IoT Testing

Prof. Dr. Ina Schieferdecker
Axel Rennoch

What are the most important challenges?



Harsh, unreliable
environments, **distribution**



Highly **dynamic**, large
numbers of different sensors,
openness, variability

Limited
resources,
scaling



IoT solutions demand new
approaches in analytical
quality assurance

What are the most important challenges?

Examples

- **Long periods** of *operation*
- *Extension* of development-related **QA** into **run-time**
- **DevOps** *links* together **testing**, run-time **monitoring** and **certification**
- Test level „**operations**“ after the usual *system tests* and *acceptance* ...
- ... considers **later changes**
 - *Extension* to interfaces,
changes to system parts,
diagnosis of **new weak points**...
- **Liability for damages** caused following misuse or security incidents.

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Quality attributes – Specific priorities for IoT systems

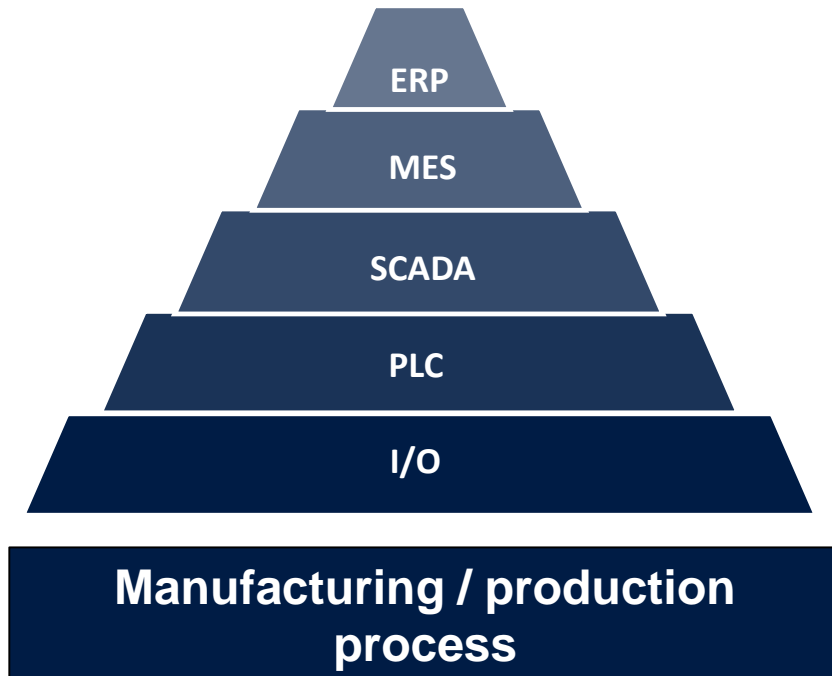
Reason	Prioritized quality attributes for IoT
Specific (distributed) architecture	Interoperability Performance and capability Adaptability Robustness and resilience
Interrelated lifecycles and interdisciplinary nature of IoT	Compatibility Maintenance Portability
Interrelated and wide ranging business processes which can be represented in IoT systems	Functional security (safety) IT-Security Privacy Usability Ethical aspects

Change to a new architectural paradigm

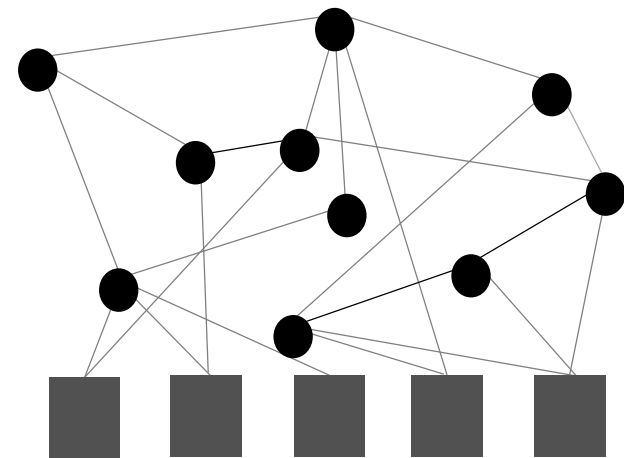
Today

Future

hierarchical



orchestrated



Open, dynamic, scalable

Thermostat



Lamps



Loud speaker

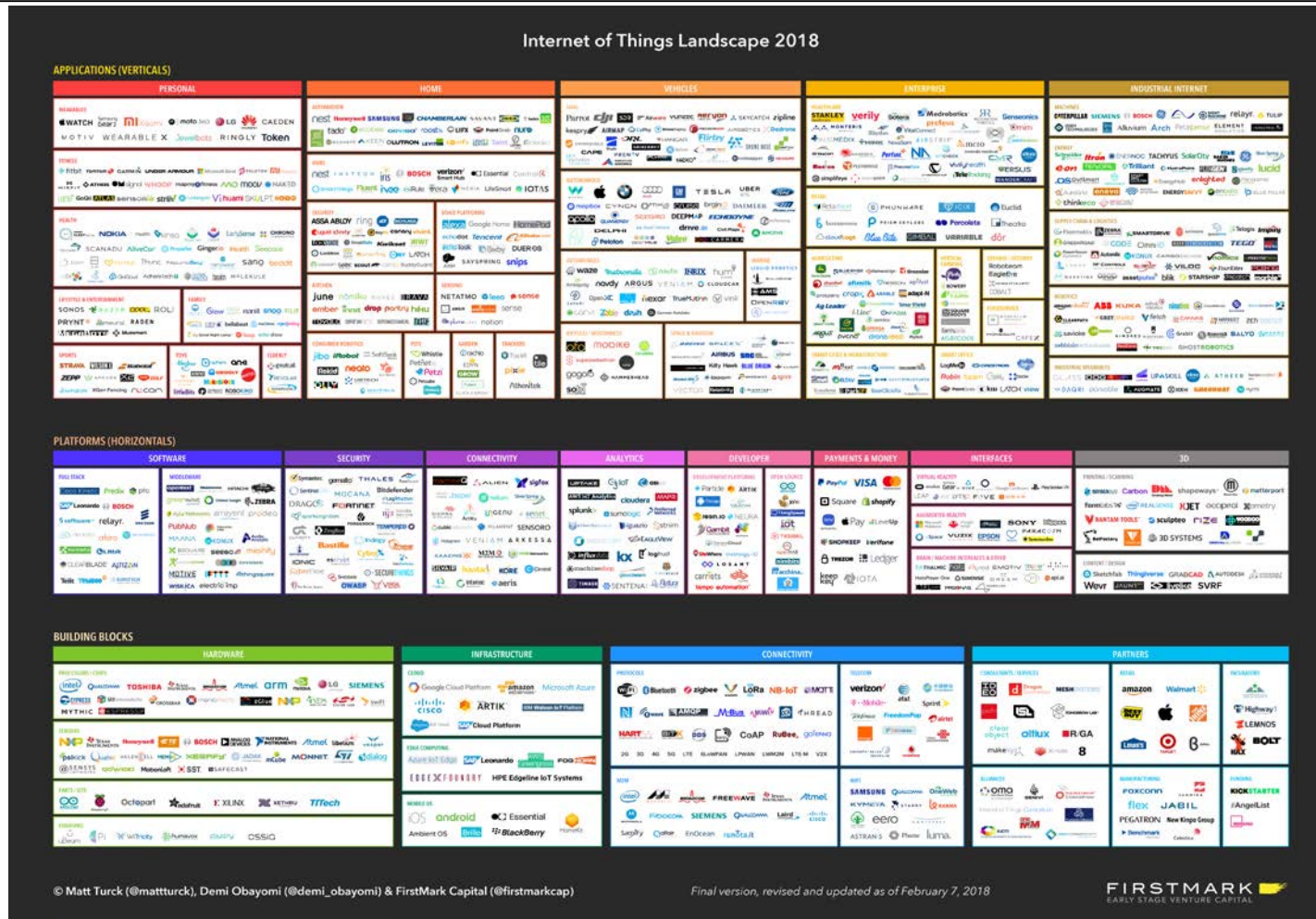


Gateway

compatibility, portability

High level of variance in testing IoT solutions

Example: high diversity of platforms



The **priority** of **test objectives**...

- relates to the priority of the **quality attributes** to be tested
- ... must be **continuously** evaluated over the lifecycle of the IoT system
- ... and where necessary adjusted or extended

Useful subdivisions of IoT test requirements and test objectives are

- Process
- System/component
- Communication protocols

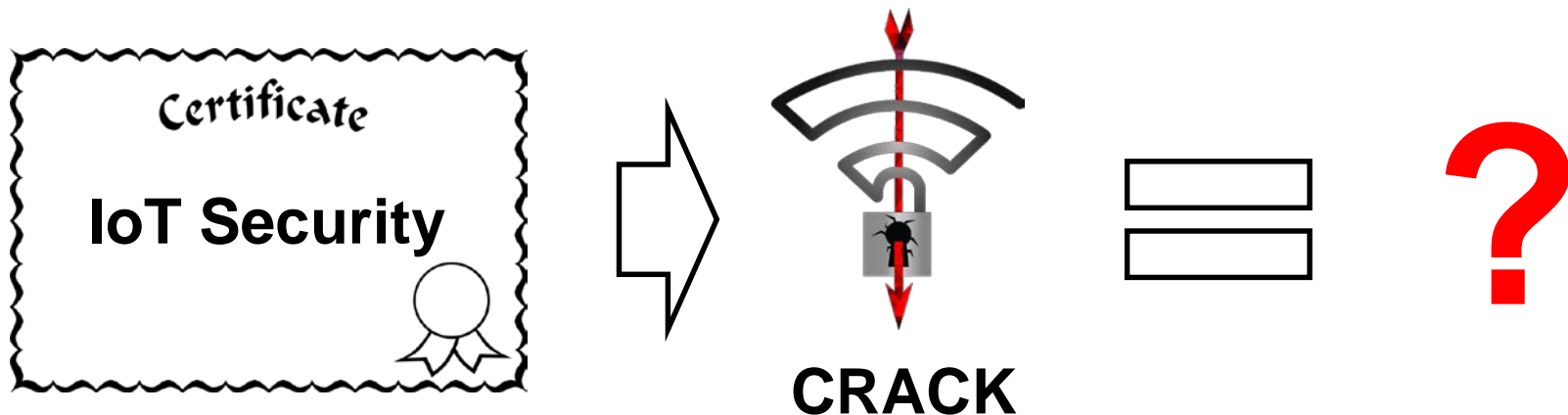
Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

- After the usual *system and acceptance tests* could follow a very long operational life. → **new test level „operation“**
- Parts of the IoT solutions may be included → **Updates + Tests during „operation“**



How do we deal with certificates after a security gap has been revealed?

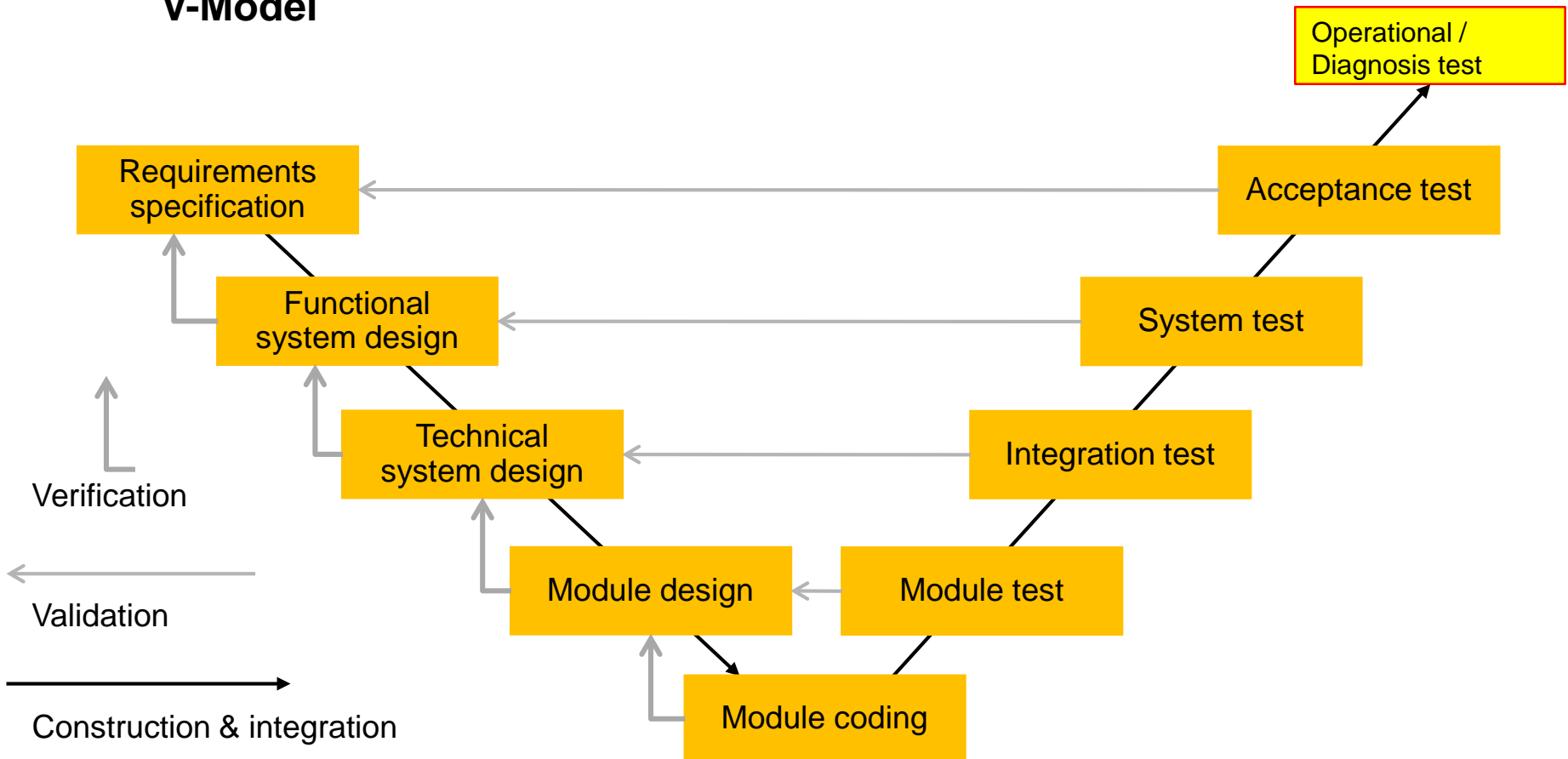
Earlier certificates issued at *product release* lose their meaning
→ New liability questions after security incidents



General test levels

V-Model

IoT specific



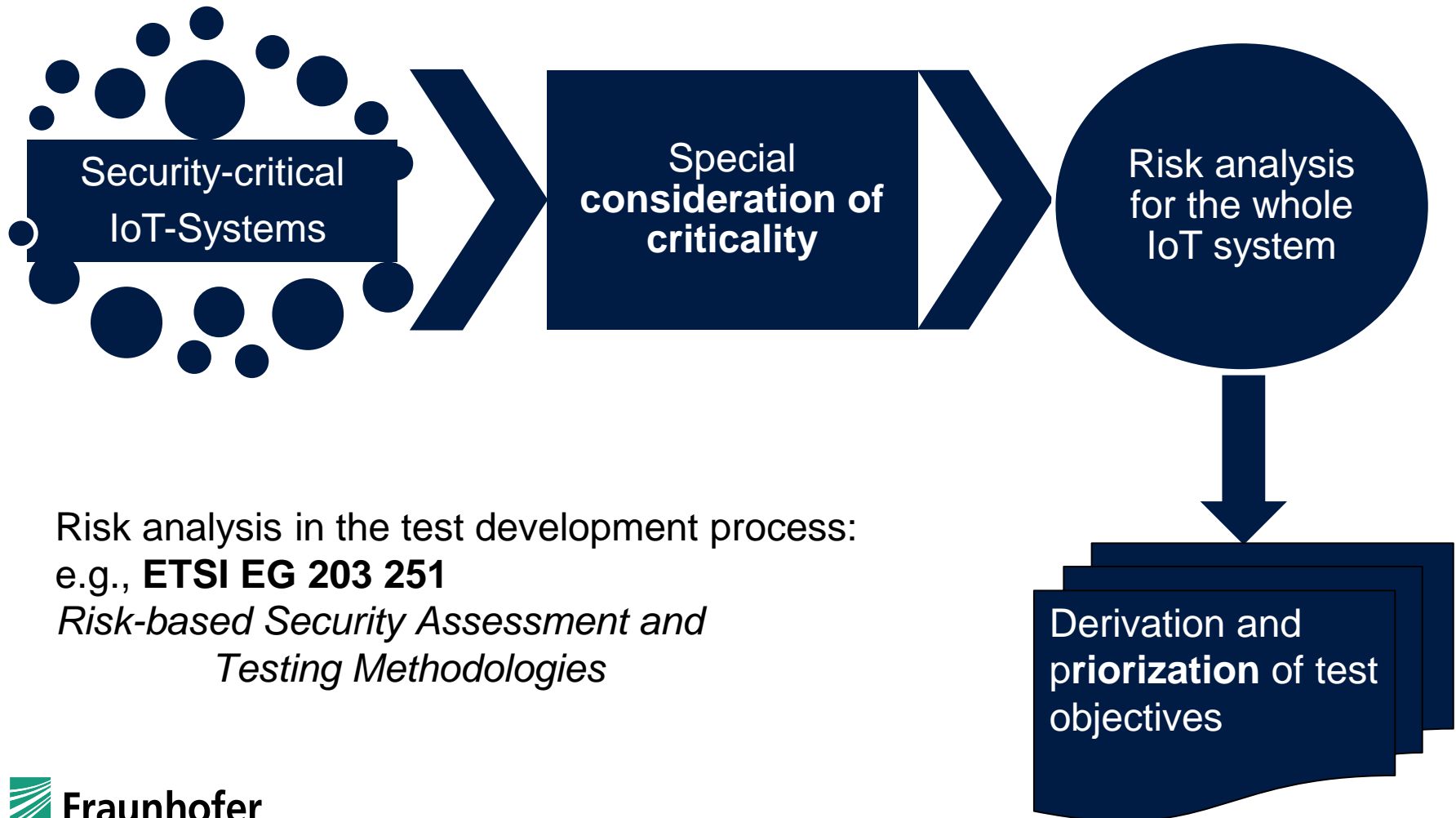
For IoT specific test levels

Test level	Example	Remarks
Acceptance test / system test or certification according to general test and integration requirements	Information security Conformity with supporting protocols Conformity with standardized procedures	Dependencies between usage profiles must be considered (e.g., <i>private vs. industrial application, military application</i>). Conformity in this sense relates to standards and standards-like documents .
Integration test for the embedding of the test object into its (test) environment	Compatibility Interoperability	Can strongly depend on the specific usage scenarios of the test objects. System environment can include high levels of complexity or possibly not fully foreseen behavior (e.g., <i>future new services</i>) Environment can also be created by simulation.

For IoT specific test levels

Test level	Example	Remarks
Operational/ Diagnosis test in the production environment and possibly also during the productive phase (e.g., <i>passive tests</i> for monitoring of behavior in operation)	Presence of required services (e.g., production tests by the manufacturer), Test scenarios for sustaining operations	Since the system environment cannot completely recreate the anticipated behavior and may change, <i>tests and analysis must also be required in operation.</i> Triggers are, for example, new deficiencies or updates that cannot be executed in a lab setting. <u>No „continuous“ tests without user permission.</u> It is possible that associated additional security risks (IT-Security and safety) must be considered.

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary



Risk analysis in the test development process:
e.g., **ETSI EG 203 251**
*Risk-based Security Assessment and
Testing Methodologies*

$$\text{Risk} = \text{Probability} * \text{Impact}$$

Example definition:

The possibility of an attackers (threat) for uncovering a vulnerability in one or more assets and the resulting damage to the organization.

(Source ISO 27000)

Process according to ISO 31000 / 2009

1. Establishing the context
2. Risk assessment
 - a. Identify risks
 - b. Analyze risks
 - c. Evaluate risks
3. Risk treatment
 - Iterative, back to 1.

Accompanying activities

- Monitoring and review
- Communication and consulting

Which vulnerability lists are openly known?

Example IT-Security (1)

Microsoft The STRIDE Threat Model

- Spoofing identity
- Tampering with data
- Repudiation
- Information disclosure
- Denial of service
- Elevation of privileges

Which vulnerability lists are openly known?

Example IT-Security (2)

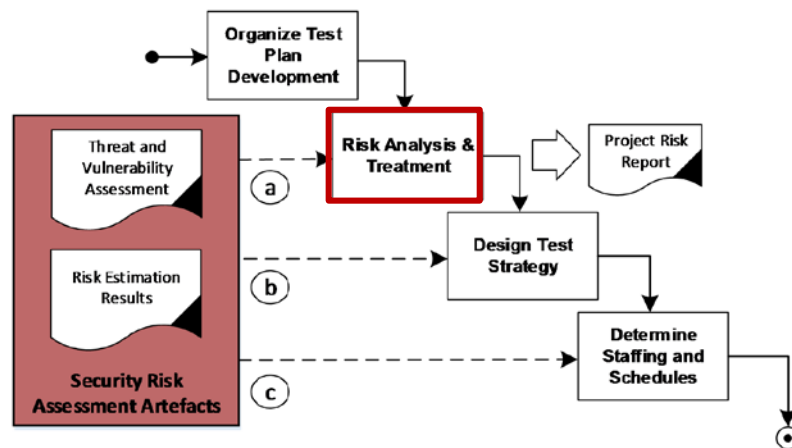
OWASP IoT Testing Guides (TOP 10)

- Insecure web interfaces
- Insufficient authentication / authorization
- Insecure network services
- Lack of transport encryption
- Privacy concerns
- Insecure Cloud interface
- Insecure mobile interface
- Insufficient security configurability
- Insecure software / firmware
- Poor physical security

Involvement of risk analysis using ETSI EG 203 251

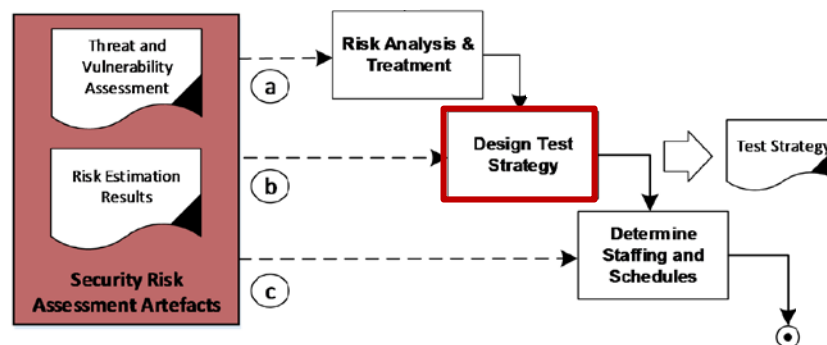
(a) Assessment of a **risk analysis**

- 1) Relevant security risks with **particular focus for security tests**
- 2) **Other product risks** or project risks relating to **missing resources**, technical problems connected to **test infrastructure**.
- 3) Develop an **overall consideration of risks** for the test project



(b) Design risk-based **test strategy**

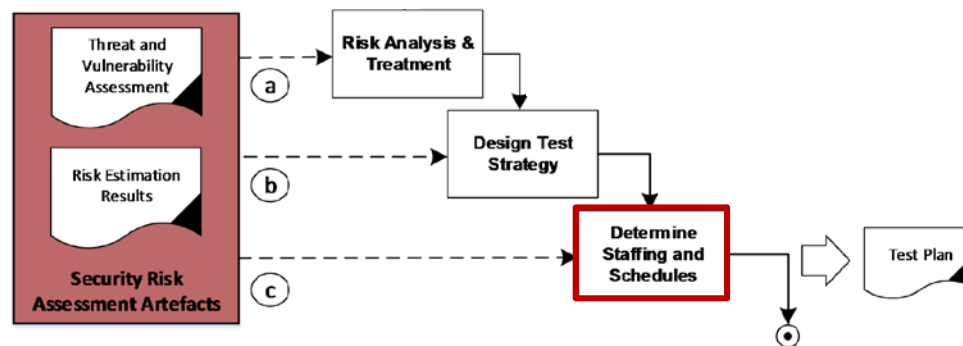
- 1) Assign **vulnerabilities** and **attack scenarios**
- 2) Identify potential vulnerabilities which have the **biggest impact** on the general security risks
- 3) Allocate **test techniques** which are suitable for uncovering the identified vulnerabilities
- 4) Assign **test completion criteria**
- 5) **Priorize** test objects and/or test conditions



Involvement of risk analysis using ETSI EG 203 251

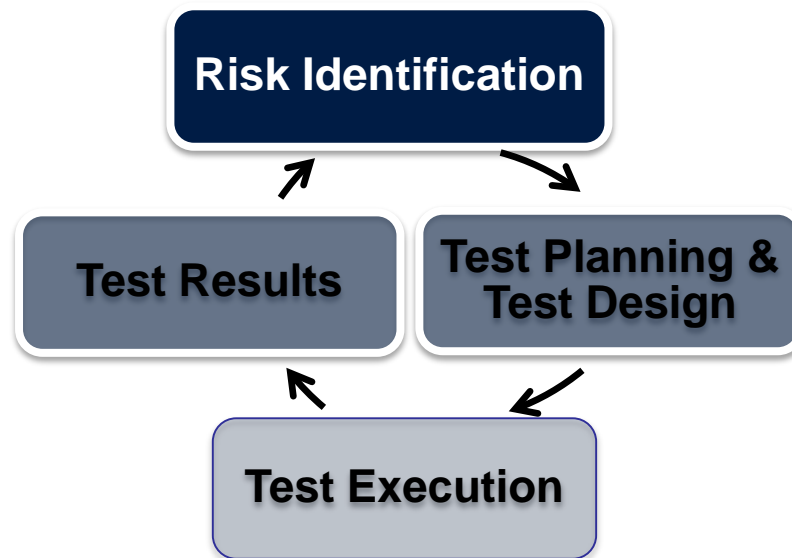
(c) Risk-based resource **planning** and test **scheduling**

- 1) Check/obtain **competency in security**
- 2) **Allocate resources** taking account of required test effort
- 3) **Create the test plans** such that the first test objects tested are those where the largest impact on the handling or minimization of identified security risks.



Risk-based testing (Example: Security)

Inclusion of test experiments



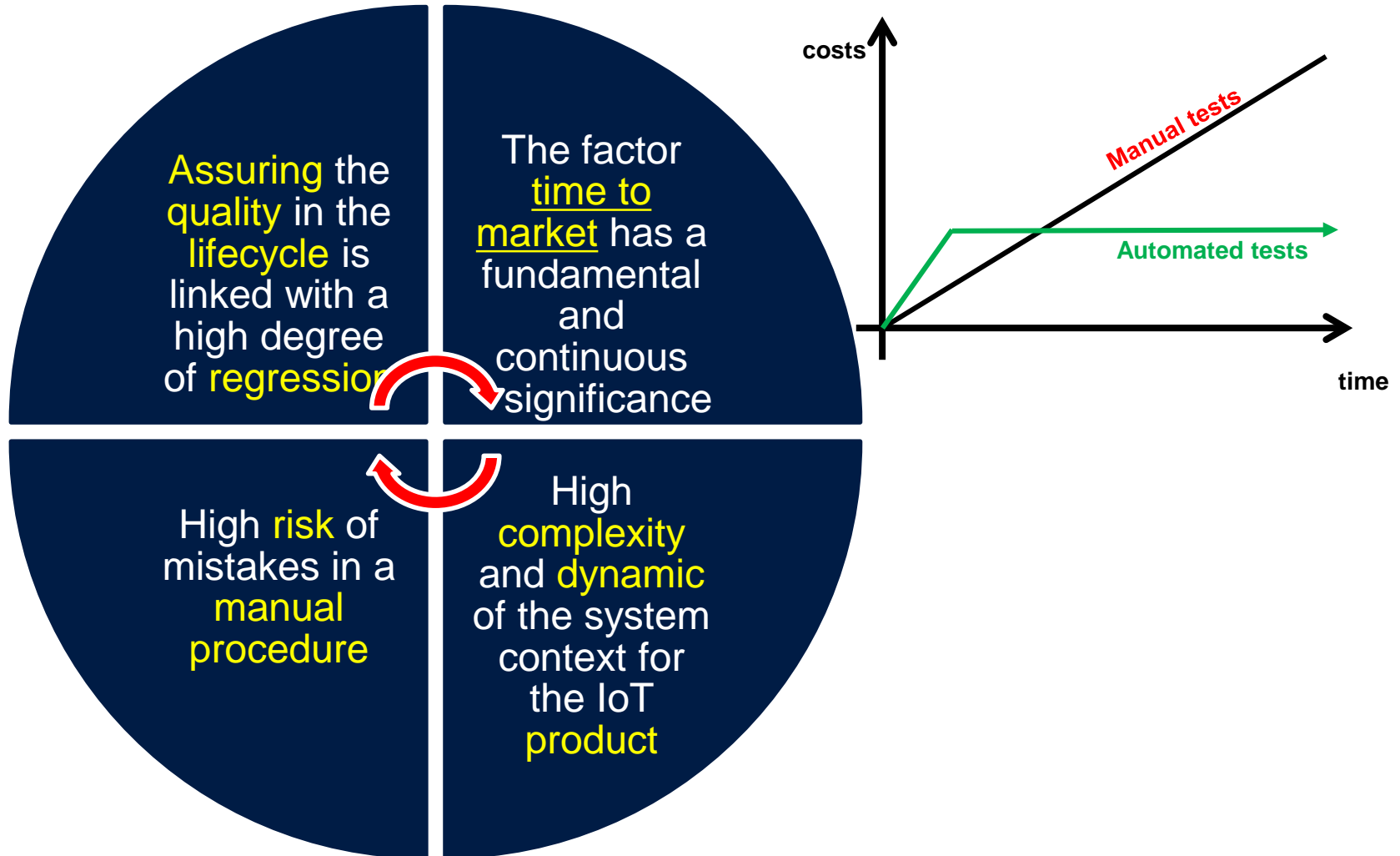
- Provide justification for the **absence** of potential **vulnerabilities**.
- Provide justification for the functional **correctness** of treatment **scenarios** and **countermeasures**.
- **Discover unknown** risk factors (i.e. vulnerabilities)
- Provide **feedback** for **reassessing risk** values

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Specialties of IoT testing

Perspective	Specialty	Test variations in addition to „classic“ software and protocol testing
Applications (analytics, visualization and control)	High level of significance for security and usability	GUI , Usability and (mobile) App Testing Performance und Scalability Testing Security Testing Crowd Testing
IoT perspective (platforms and interfaces, computation-, aggregation- and storage services)	High level of significance for security , conformity / interoperability and data quality	Real-time testing GUI testing (for management software) Security testing
Physical perspective (devices and device connectivity)	High level of significance for security , conformity / interoperability and availability High level of significance for robustness , physical security , <i>resource</i> usage	Performance and scalability testing Service testing (connectivity) Security testing Embedded Systems Testing Robustness -Testing (Physical) Security Testing Performance-Testing (Resources)

Need for test automation



Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Which IoT test architectures are there?

■ **Device-based** IoT test architectures

- E.g., for testing **Retroboxes** or **Gateways**

■ **Service-based** IoT test architectures

- E.g., for the data-oriented testing of **Dashboards** in the **Cloud**

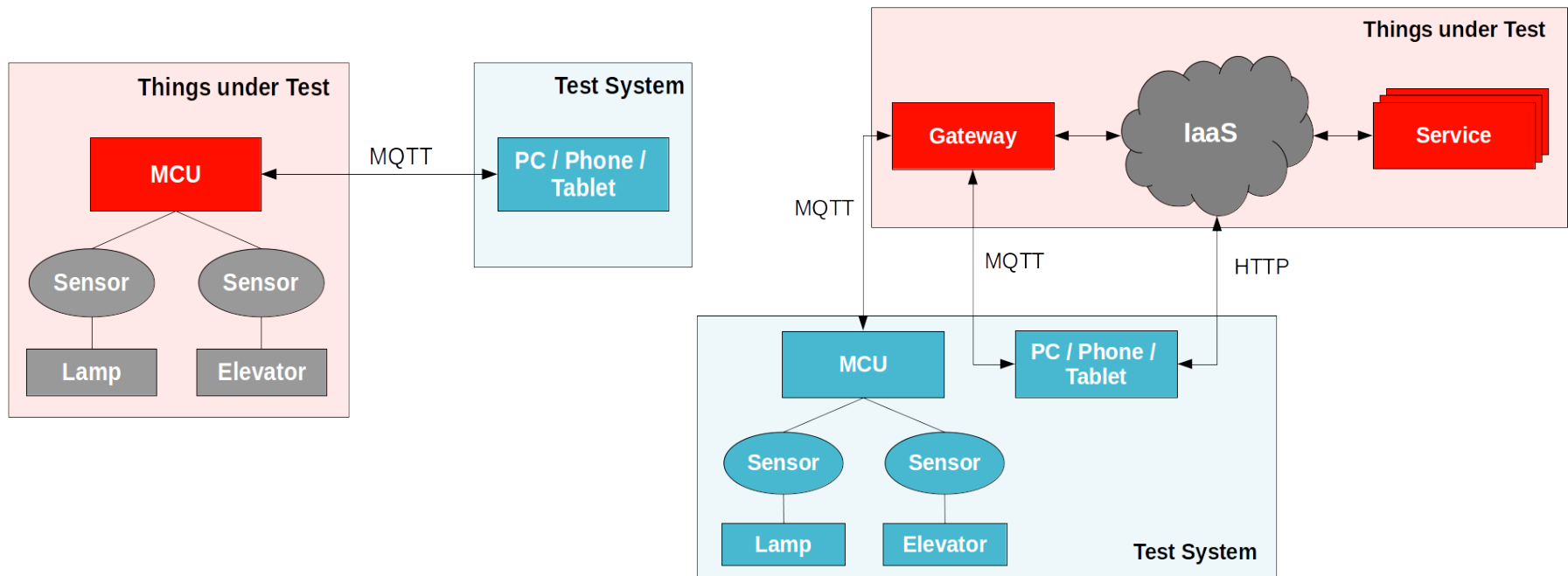
■ **Infrastructure-based** IoT test architectures

- E.g., for testing **oneM2M** functional elements

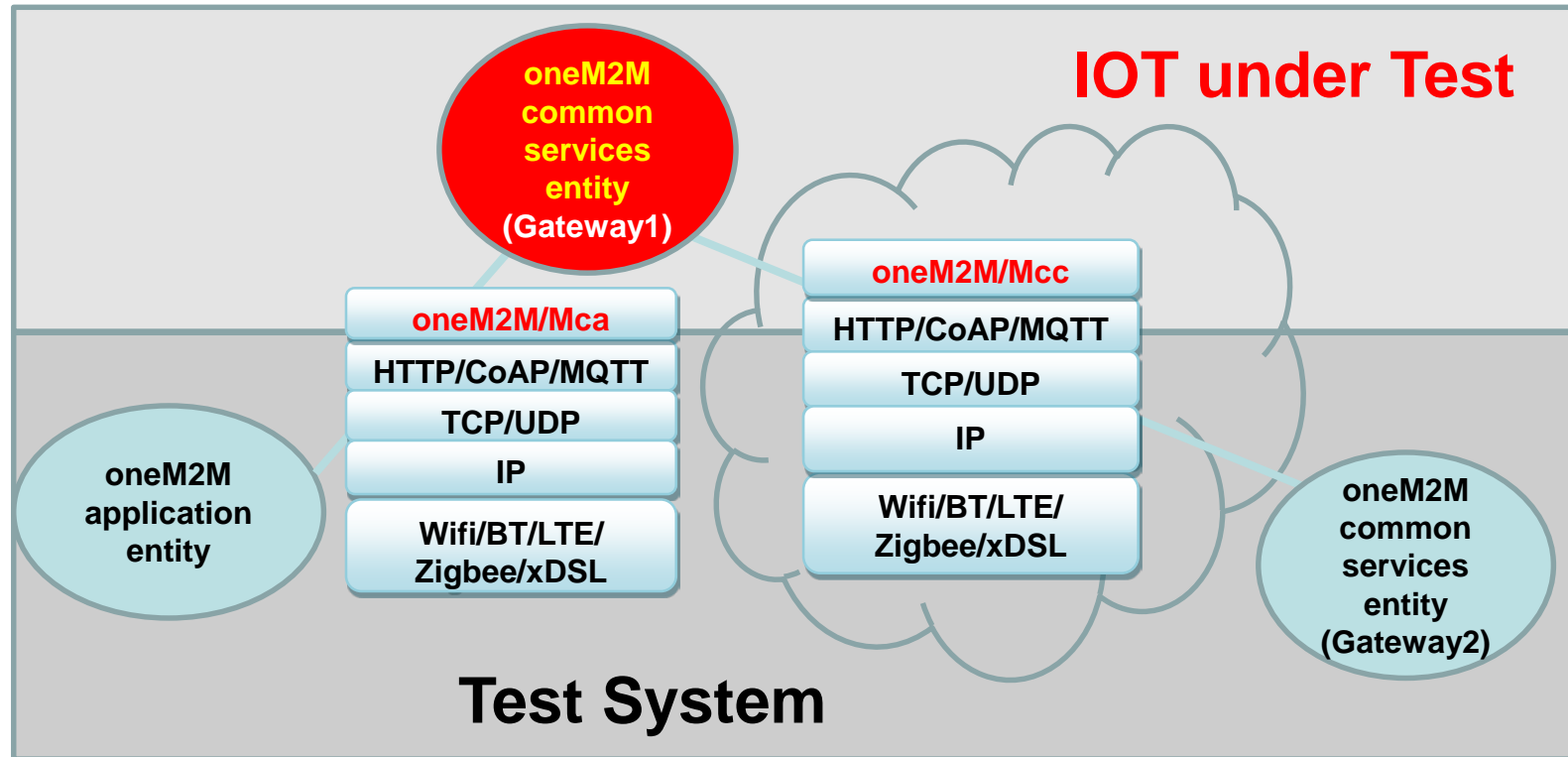
Since IoT systems are distributed, many **distributed test architectures** and **corresponding process strategies** can be applied.

Example: Efficiency improvements by ***virtualization of the entire test system***

Example: various test systems (1)



Example: various test systems (2)



Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

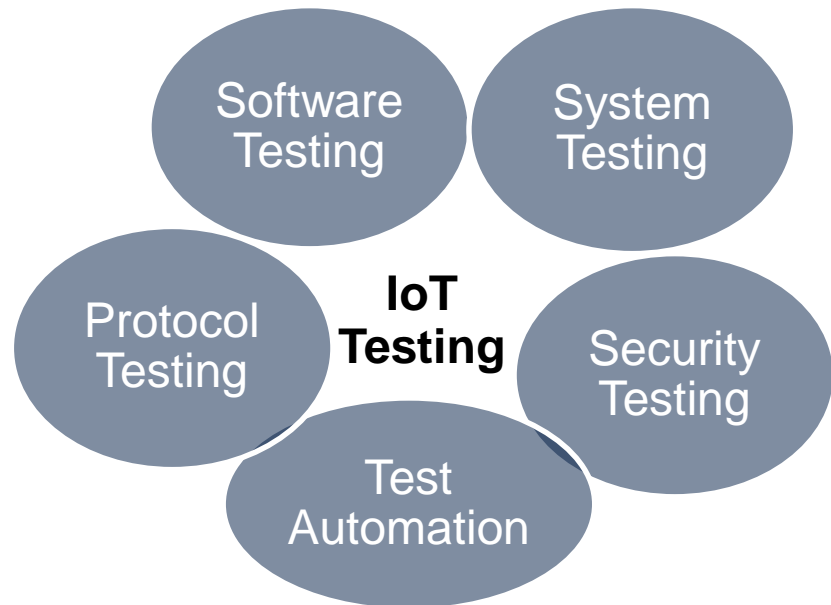
Testing methods and testing tools are **NOT** fundamentally new

they are a **special selection** of *established approaches*

- considering the *IoT-specific characteristics* of the SUT
- considering the *specifics from the requirements* analysis

Main focus for practical applications

- Interoperability
- Security
- Performance



Where do we need testing techniques?!

Some examples

Familiar test objectives

- **Protocol stacks**
 - IETF-based: CoAP, MQTT, etc.
 - IEC-based: OPC-UA
 - ITU-based: M2M
- **Application frameworks**
 - Eclipse: Kura, Scada, etc.

Additional test objectives

- **Security**
 - ISO: common criteria
 - Mitre: CWE list
 - Others
- **Data**
 - Semantic real-time data

- **Protocol Test**
 - Conformity
 - Interoperability
 - Performance
- **Software Test**
 - Component testing
 - Integration testing
 - System testing
- **Test of IT-Security**
 - Risk-oriented testing
 - Fuzz testing
 - Online testing
- **Data quality**

Model-driven analysis strategies and **model-based testing**

optimal analytical
approaches

Definition of the SUT
interfaces

Important Best Practice
for IoT Tests

Online MBT

Test automation

Meaningfully extended
by **manual tests**

In particular
exploratory tests

Standardized test descriptions for *specific domains* and their *protocols*

- e.g., ETSI in the *telecommunication*, *automotive* and *Autosar* domains
- typically using **formal descriptions techniques**.

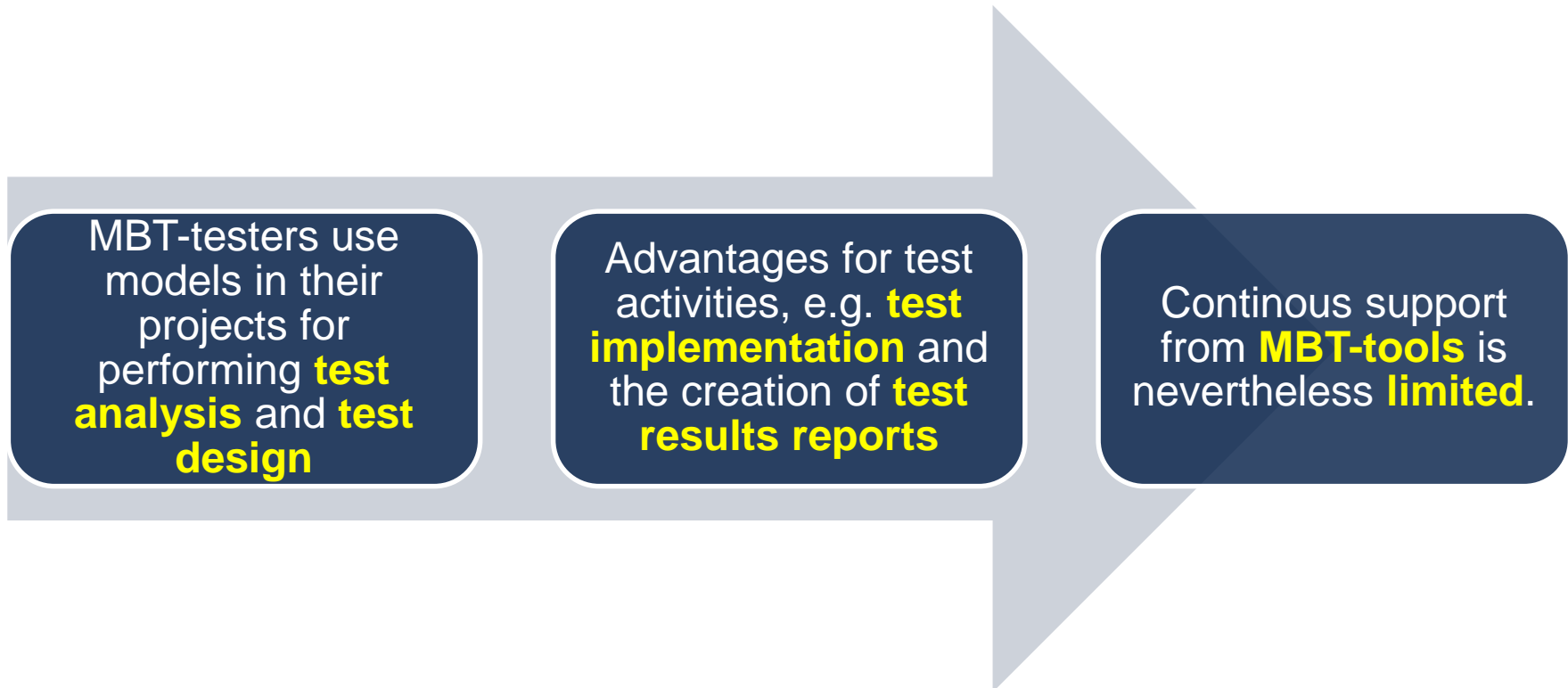
Challenges of test automation:

TTCN-3 – Testing and Test Control Notation

- **Example of a formal description**
- **International standardized test notation**, which is especially developed for the description of test scenarios
- **Test technology** which is suited to all testing techniques
 - Distributed
 - Platform-independent
 - Extendable
 - Adaptable to the environment

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

... and why MBT?



Model-based testing is an innovative approach for improving the effectiveness and efficiency of the test *generation* process.

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Reminder: Security Tests are extremely significant!

Why is IT-security often not considered in sufficient depth – especially for IoT projects?

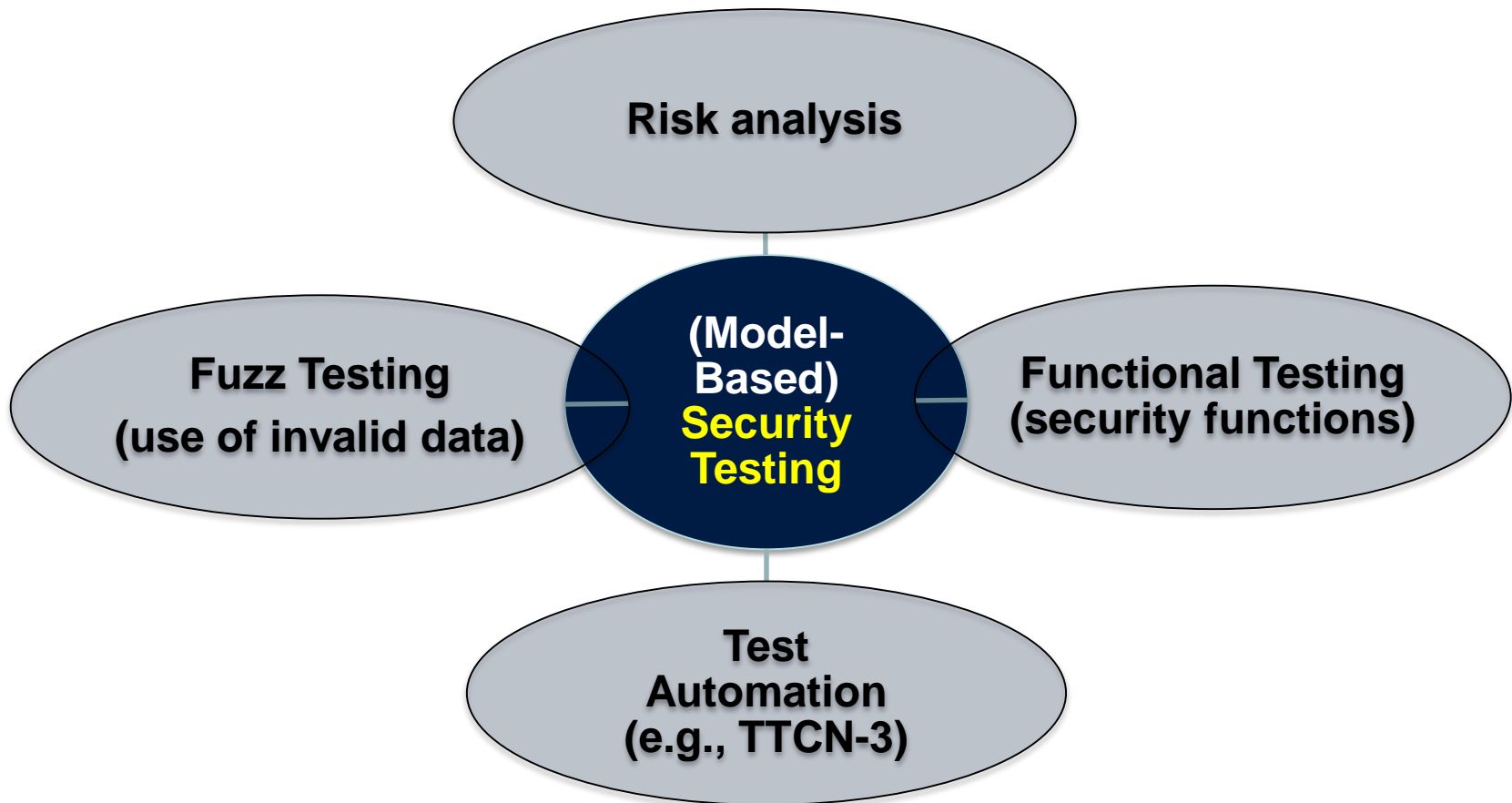


** IoT Systems are composed of a wide range of different components which are frequently developed by separate teams*

Example: Botnets

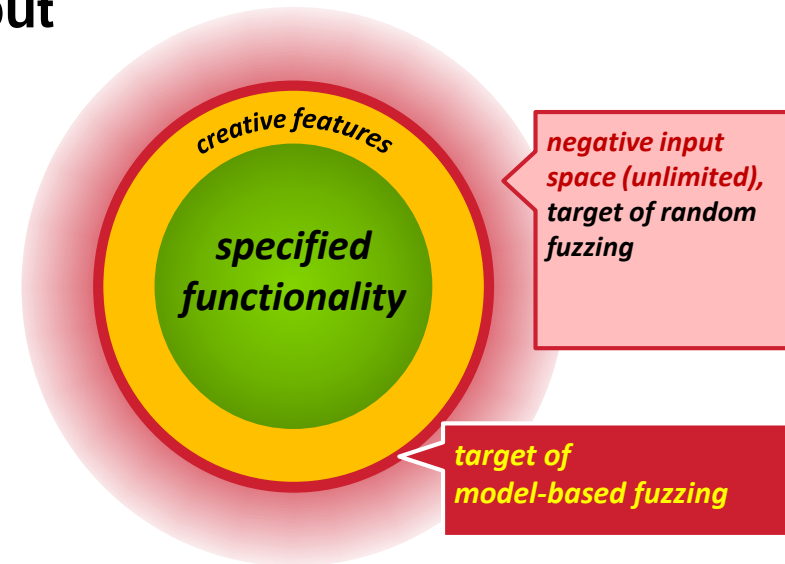


- Group of **automated programs (malware)** to impact system security which run on **interconnected computers**
- Without the consent of the owner



Ina Schieferdecker, Model Based Security Testing: Selected Considerations (Keynote) Sectest 2011, Workshop on the 4th IEEE International Conference on Software Testing, Verification and Validation Berlin, Germany

- Fuzzing *originally* describes the **random generation of test vectors**
- Fuzzing is about **injecting invalid or random inputs** in order
 - to reveal unexpected behavior
 - to identify errors and expose potential vulnerabilities
- Ideally, fuzzers generate **semi-valid input data**, i.e. input data that is invalid only in *small* portions.
- **Challenge:**
Finding data that is as *invalid* as possible *and* is still *accepted*.



Test focus for security tests

Perspective	Attack Vector	Methods
Applications (analytics, visualization and control)	<ul style="list-style-type: none"> - Mobile applications - Web applications - Data and control flows 	<ul style="list-style-type: none"> - Test of web vulnerabilities - Test of sensitive data in mobile devices - Data flow analysis / Proxy / Man in the Middle-Attack - Denial of Service - Search for logical vulnerabilities in the overall concept
IoT perspective (platforms and interfaces, computation, aggregation and storage services)	<ul style="list-style-type: none"> - Cloud services - Web interfaces (configuration interfaces) of devices - Data and control flows - Access and rights management - Update mechanisms (over the air updates) - Localization service 	<ul style="list-style-type: none"> - Test of web vulnerabilities - Data flow analysis / Proxy / Man in the Middle-Attack - Search for protocol vulnerabilities or incorrect configurations e.g., unencrypted communications - Denial of Service - Search for logical vulnerabilities in the overall concept - Spoofing of end devices
...		

Test focus for security tests

Perspective	Attack Vector	Methods
...		
Physical perspective (devices, device connectivity)	<ul style="list-style-type: none"> - Backend APIs - Data and control flows - Encryption - Other communication between the IoT Layer and the Network Layer - Device storage and storage extensions (e.g., SD-cards) - Device firmware - Physical device interfaces - Device network interfaces - Physical manipulation or theft of the device 	<ul style="list-style-type: none"> - Data flow analysis / Proxy / Man in the Middle-Attack - Search for <i>logical</i> vulnerabilities in the overall concept - Test of web vulnerabilities - Making use of known protocol vulnerabilities - Denial of Service - Search for sensitive data (passwords, keys,...) and manipulation of data - Analysis of firmware - Back channel attacks

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

What role does interoperability play in IoT ?

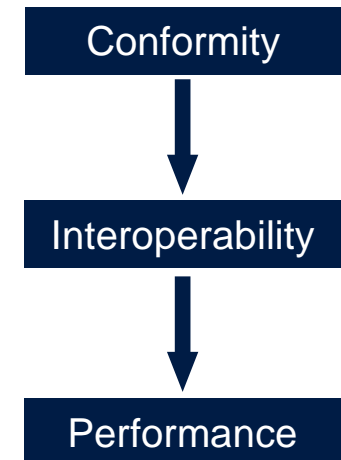
- IoT-Systems and their components can differ **strongly different** and they can originate from **different manufacturers**.
- Example:
 - **Mass-produced** sensors and actuators
 - Specially configured **gateways** as well as other devices with **dedicated protocols / versions**
 - Wide variety of **end devices** for **display** of analyzed data.
- Example Smart Home:
LED-Lamps, Amazon *Echo*, *iPhone*, switchable intermediate *plugs*, Telekom-Starter Package, window *sensors*, Smart Home *heating* package, ...

Are the various systems and components able to interact with each other?

What distinguishes interoperability tests?

Interoperability tests evaluate the ability of the software product to ***interact with one or more specified components*** or systems

- **functional** test
- **data exchange** between two selected systems (client / server)
- Normally the interoperability tests are executed after successful **completion of conformity tests**
- **correct** inputs according to the protocol
- *Robustness* tests using deliberately **defective data** which tests the stability (reaction) of the system



Test object	Technique	What is tested?
Technical interoperability	Basic tests of connectivity and communications protocols	Coupling of hardware / software components in order to ensure <i>basic communication</i>
Syntactic interoperability	Targeted checks of messages and the syntax of abstract data formats . Use of encoders and decoders	Correct use of syntax for e.g., HTML, XML or ASN.1 data structures
Semantic interoperability	Execution of sample scenarios and user scenarios , possibly with support from standardized Use Case catalogs	Checks on whether the implementation of the interconnected components/systems follows a common interpretation .

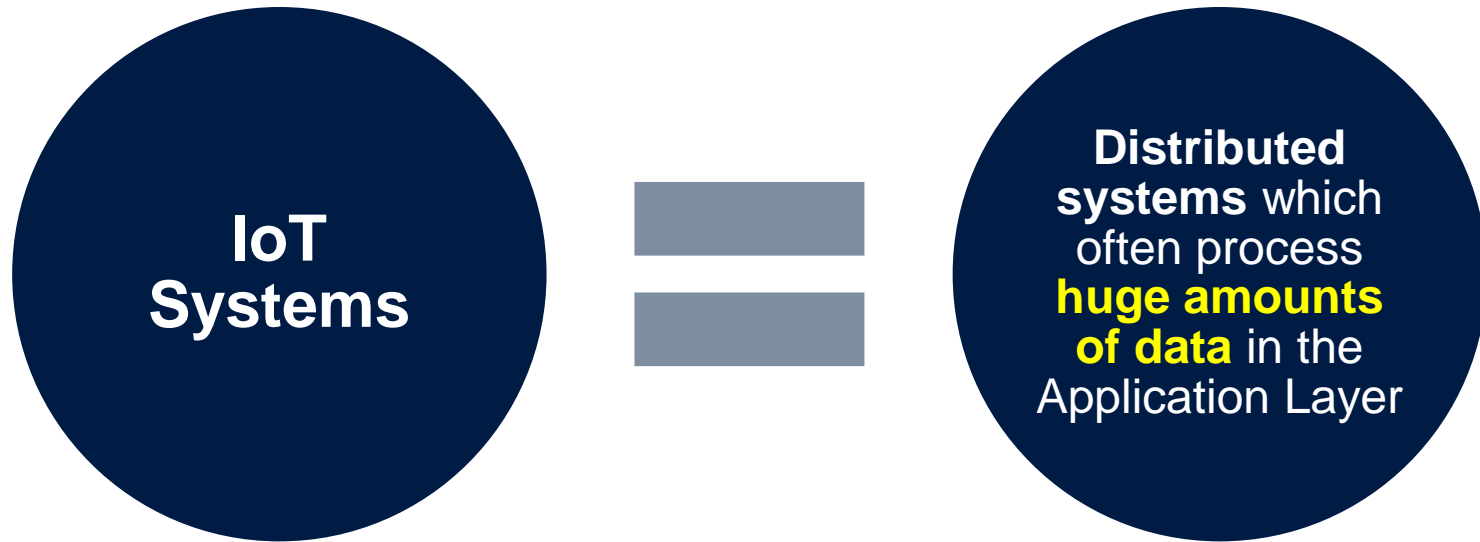
☐ Event Format

- ☐ Conventions: **Manufacturers** of electronic **equipment** or **software**
- ☐ Interoperability test of a product with **products** from other manufacturers
- ☐ Test on the basis of a (standardized) **catalog of test objectives**

☐ Catalog of test objectives

- ☐ Tabular form
- ☐ specification of **configuration**, sequential flow of **triggers**
- ☐ specification of **participating components** or systems

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

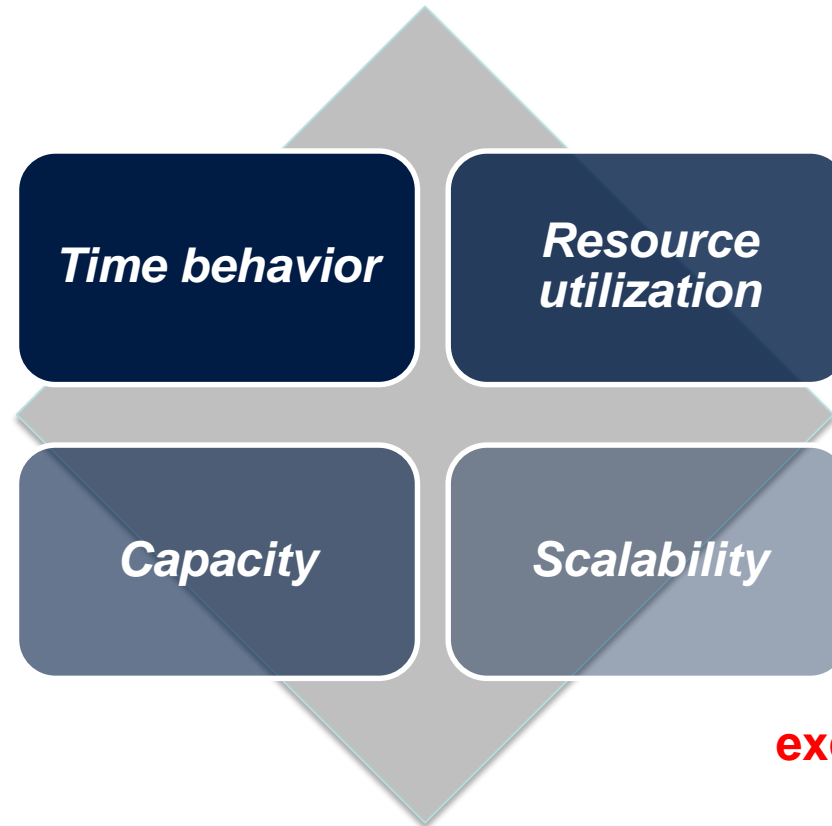


Essential aspects for quality assurance:

- A **suitable architecture** must be selected (e.g., with the use of elements from Edge Computing) to enable the *efficient processing of heterogeneous data streams*
- **Scalable** performance of each IoT **component**
- **Monitoring** and test of performance **in operation**

Reminder about performance tests

Several types of performance
to test



under **normal** and
exceptional conditions

Automation (required):

- **High data volumes**
(transaction volumes)
- **Distribution** of test
interfaces

Parameterization (usual):

- **Number of devices**, Client/Server-
relations, possibly specific
infrastructures
- **Time intervals**,
environmental *conditions*
- Specific **boundary conditions**
(*decision* criteria)

What needs to be considered for performance tests?

The **configuration** of the test system:

- **Many *interfaces***
- **Distribution** and ***synchronization*** of the scenario control across all *test components*
- **Manufacturer-independent Use Cases for test** (standardization!)

Specialties for **IoT**:

- Simulation and reaction from **devices/sensors** *instead of users*
- *Irregular/ uncommon* **status changes** (caused by environment)
- **Unsteady connectivity**
- Interaction of *various* communications *protocols*
- Consideration of specific **hardware characteristics**
- **Delays** in *Cloud transfers* (Test access points)

What factors distinguish the test tools?

Combination of
protocol test and
application **test tools**

Consideration of (real-
time) company **data**
analysis (data mining,
statistics, etc.)

High level of
abstraction for the **test**
description language
(various transfer layer)

Re-use of conformity
scenarios for load test
scenarios

Simulation over **long**
time periods and
verification of **large**
volumes of collected
data

→ **More effort for the configuration and simulation of the test environment!**

Overview of tools for performance testing

Approach	Tool example	Explanations
<u>Dynamic Analysis</u>	Dynamic <u>analysis</u> tools uncover defects which can only be revealed <i>when the program is running</i> (e.g., time dependent defects and memory bottle-necks)	These are typically used for the component test and component integration tests , as well as for tests of middleware.
<u>Performance test</u> , <u>Load test</u> , <u>Stress test</u>	<p>Performance test tools <u>monitor</u> and protocol how a system responds under <i>various simulated usage conditions, e.g the number of parallel users, ramp-up behavior, frequency and relative proportions of transactions.</i></p> <p>The load is created by simulating virtual users which execute a selected group of transactions. <i>These are <u>distributed</u> over various test machines which are generally known as load generators.</i></p>	<p><i>Software test</i> with expected and extreme loads submitted to a running system (e.g. for simulating event data).</p> <p>The behavior of the system is then observed and analyzed.</p>
<u>Monitoring</u>	Test <u>monitors</u> continuously analyze, verify and <u>record</u> the usage of specific system resources and issue warnings of possible problems in providing services.	


Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Product certification

=

Checking conformity to the requirements stated in standards and additional normative documents (DIN EN ISO/IEC 17065)

 Basis for the certification: guiding principles, **norms** and **standards**

 Main focus for IoT:
Checking IT-Security

☐ Collection of **high-level norms** that are still **too immature**

☐ Currently there is **no** „IoT Standard“

☐ Checking *products, processes and services*

☐ *Functional security requirements, stability*

☐ Conformity and vulnerability to errors in **communication protocols** typically used in IoT

The goal is to establish confidence in a part of the IoT

■ Fundamental questions in a certification

- ☐ Which **criteria** shall be applied? test, evaluation and checking approaches?
- ☐ Which **part of an IoT product** or an IoT solution shall be included?
- ☐ How shall the **checks** be performed and what must the **certification authority** achieve?
- ☐ General validity / **level of detail of requirements**:
check/test(pass) criteria

■ Practical questions

- ☐ What shall be the certificate's **period of validity**?
- ☐ What happens if a security **incident** occurs?
- ☐ What significance do **updates and patches** have for the certificate?

Wide *range of varieties* of IoT-Devices
and IoT-Services

- Major significance of certification

Test lab

- Provides **results** from **inspection** and **checking**

Certification authority

- Evaluates **results** from **inspection** and **checking**
- Issues a **certificate** or a recognized **seal of approval**

Test- objectives	Test levels	Risk analysis
Test automation	Test- architectures	Test techniques
Security, Interop., Performance	Certification	Summary

Test Objectives

For all layers of the architecture

Inclusion of risk analysis

Priorization!

Test Levels

Test and Monitoring
„in operation“ due to longer operational lives and updates

Automation

Thorough test

Regression

Distributed architecture

High data volumes

IT-Security

Overall consideration

For all layers of the architecture

Many techniques

Certification

Missing standards

Scarcity of generally accepted check criteria

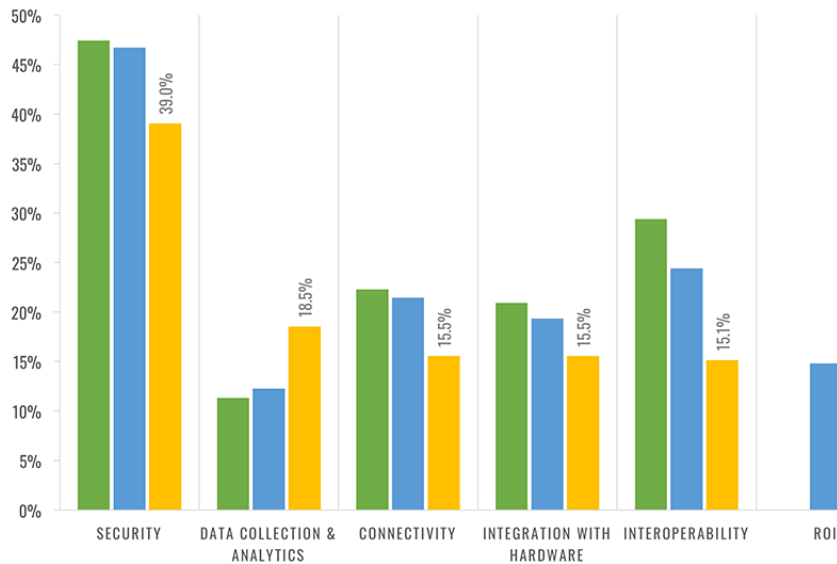
THE IOT-TESTWARE PROJECT



The Testware

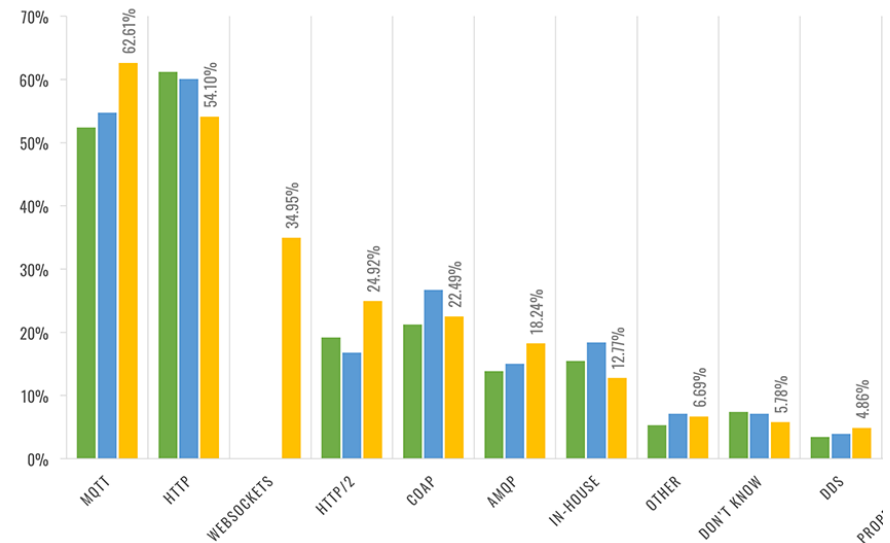
TRENDS IN IOT

TOP IoT CONCERNS / TRENDS 2016-2018



Copyright (c) 2018, Eclipse Foundation, Inc. | Made available under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0)

MESSAGING STANDARDS - TRENDS



Copyright (c) 2018, Eclipse Foundation, Inc. | Made available under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/) (CC BY 4.0)

CHALLENGES IN IOT

Wide portfolio of competences required

- Devices (sensors, HW, embedded SW)
- Platforms (Cloud, platform domain knowledge)
- Applications (SW, dashboard, business logic)

IoT platforms

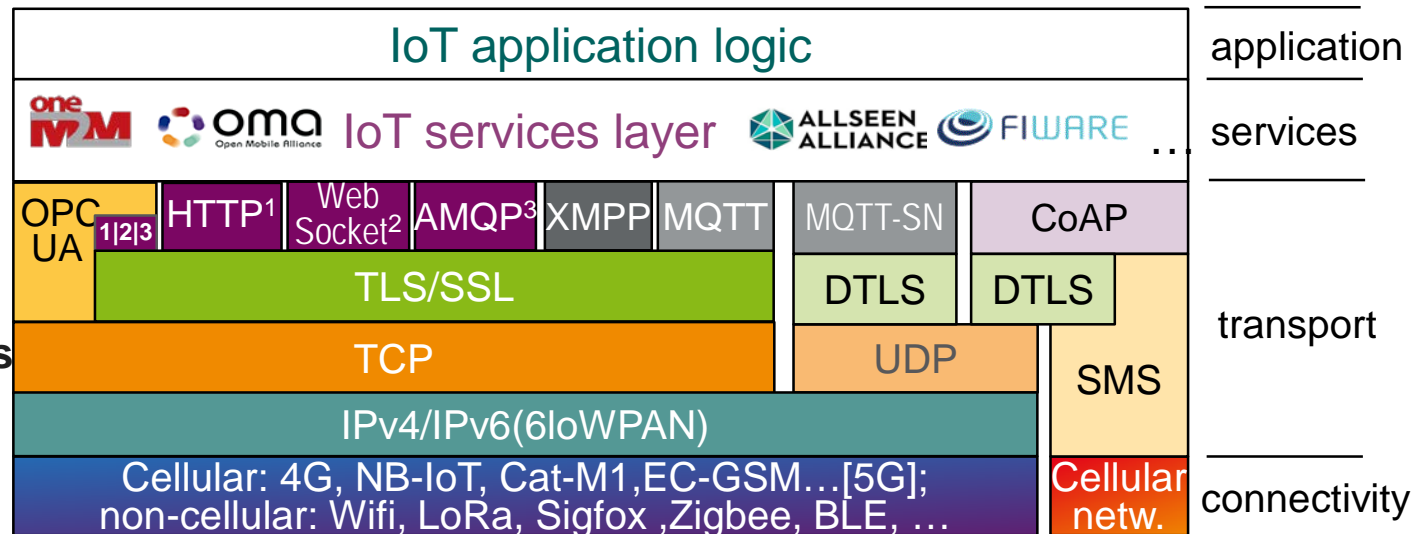
- 360+ worldwide

IoT protocols

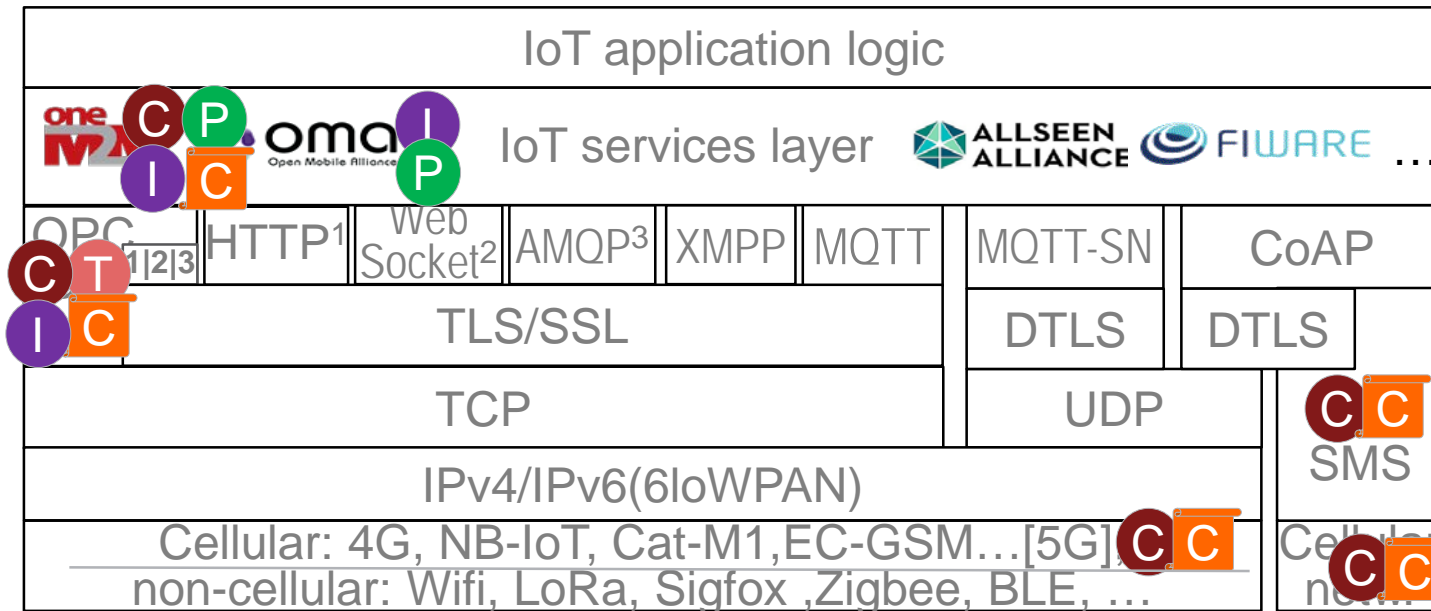
- Rich selection
- IP-based
- non-IP based

Connectivity options

- Throughput
- Latency
- Power efficiency
- Packet size



TEST COVERAGE BY SDO-S



C Conformance tests

I Interoperability tests

P PlugFest/Interoperability test event

T Compliance tool

C Certification

THE ECLIPSE PROJECT

- Supplement to running and active Eclipse projects
 - Paho, OM2M, Titan...
- **New project** at Eclipse Foundation:
<https://projects.eclipse.org/projects/technology.iotestware>
 - TTCN-3 test suites for **CoAP, MQTT, OPC-UA**, LoRa?
- Assured **licenses** for users
- **Currently in cooperation with**
relayr GmbH, Ericsson, LAAS/CNRS, itemis AG,
Spirent Communications, Easy Global Market, Iskratel/Sintesis, ...



IOT-TESTWARE

Take **available** software and tools ...

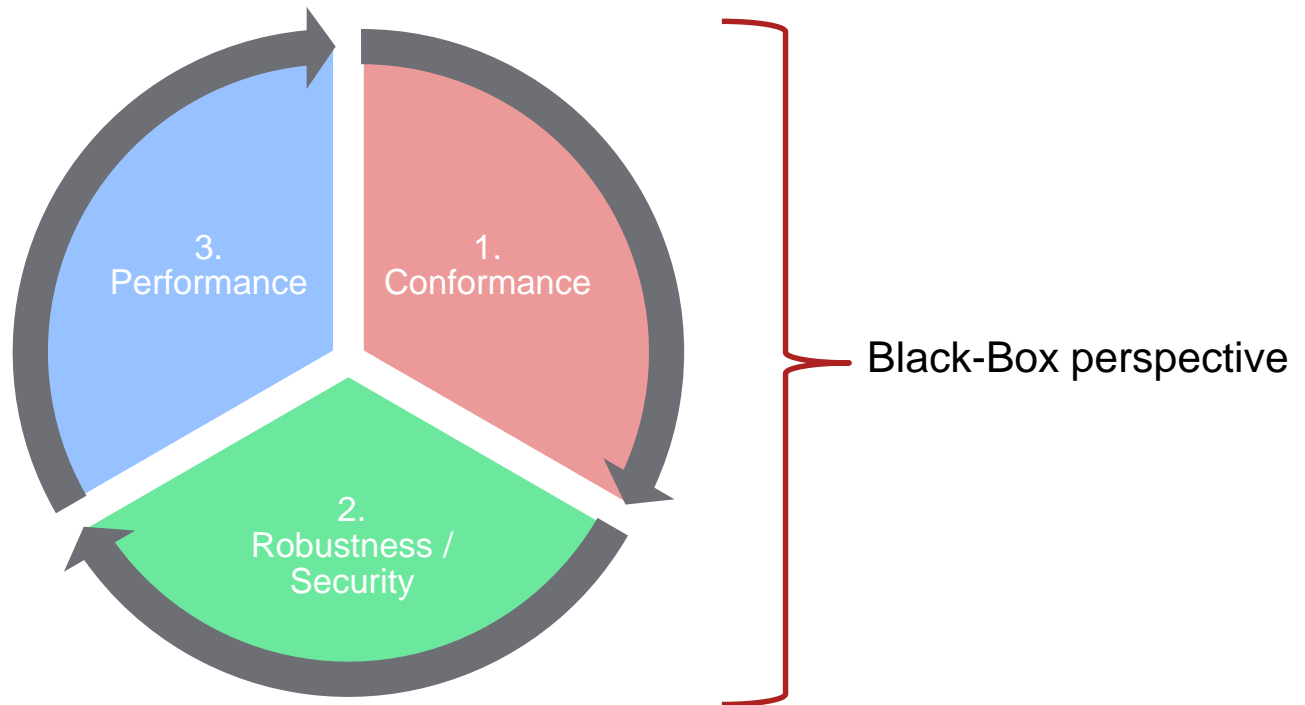


... and adding public testuites as a result of insights from IoT testing:



<https://projects.eclipse.org/projects/technology.iotestware>

IOT QUALITY



BACKGROUND

Test Description Language

- Design, documentation, representation of formalised test descriptions
- Scenario-based approach

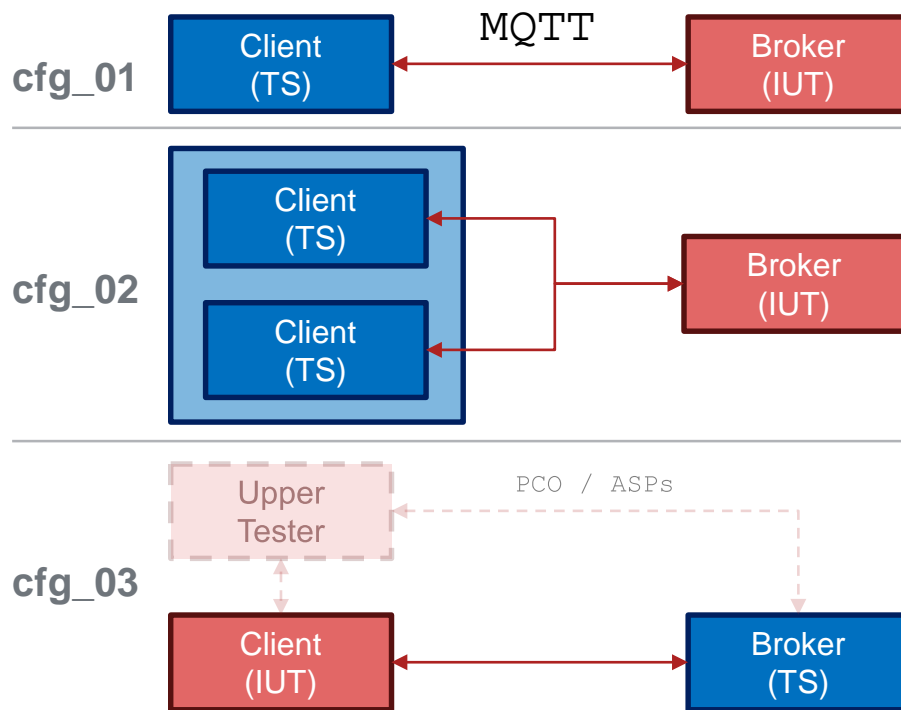


Testing and Test Control Notation

- Specification and implementation of all kinds of black-box tests
- Component-based approach



SAMPLE TESTSUITE STRUCTURE: MQTT



– Broker as SUT

- All mandatory **message data** fields
 - Regular and illegal data (Fixed/variable header, payload)

– Protocol features

- Connect/disconnect (session)
- Subscribe/unsubscribe
- Immediate publish
- Last will and Testament (LWT)
- Heartbeats keepAlive values
- Topic
- Error handling

– Client as SUT

– ...

```
Test Purpose {  
TP Id TP_HELLO_MSG_SERVER  
  Test objective  
  "Establishing..." /*Summary*/  
  Reference  
  "OPC-UA,Part-6-Mappings..."  
  Expected behaviour  
  ensure that  
    {when {...}  
    then {...}  
    }  
}
```

- **Informal text** specification possible
- Support **simple** description **structure** (*event occurrence sequences*)
- Global **keyword** definitions (*domain specific*)
- Focus on a **single test** observation for pass/fail verdict **criteria**
- ...

TEST DEVELOPMENT SAMPLE: MQTT TEST CATALOGUE

✓ Test configurations

✓ Test Suite Structure

✓ Test purpose (catalogue)

✓ Test implementation (TTCN-3)

TP Id	TP_MQTT_Broker_CONNECT_001
Test Objective	The IUT MUST close the network connection if fixed header flags in CONNECT Control Packet are invalid
Reference	[MQTT-2.2.2-1], [MQTT-2.2.2-2], [MQTT-3.1.4-1], [MQTT-3.2.2-6]
PICS Selection	PIC_BROKER_BASIC
Initial Conditions	
Expected Behaviour	
<pre>ensure that { when { the IUT receives a CONNECT message containing header_flags indicating value '1111'B; } then { the IUT closes the TCP_CONNECTION } }</pre>	
Final Conditions	

IOT TEST LANGUAGE

Did you know that **YOUR PHONE...**



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**

- *Conformance and functional testing*
- *Interoperability and integration testing*
- *Real-time, performance, load and stress testing*
- *Security testing*
- *Regression testing*

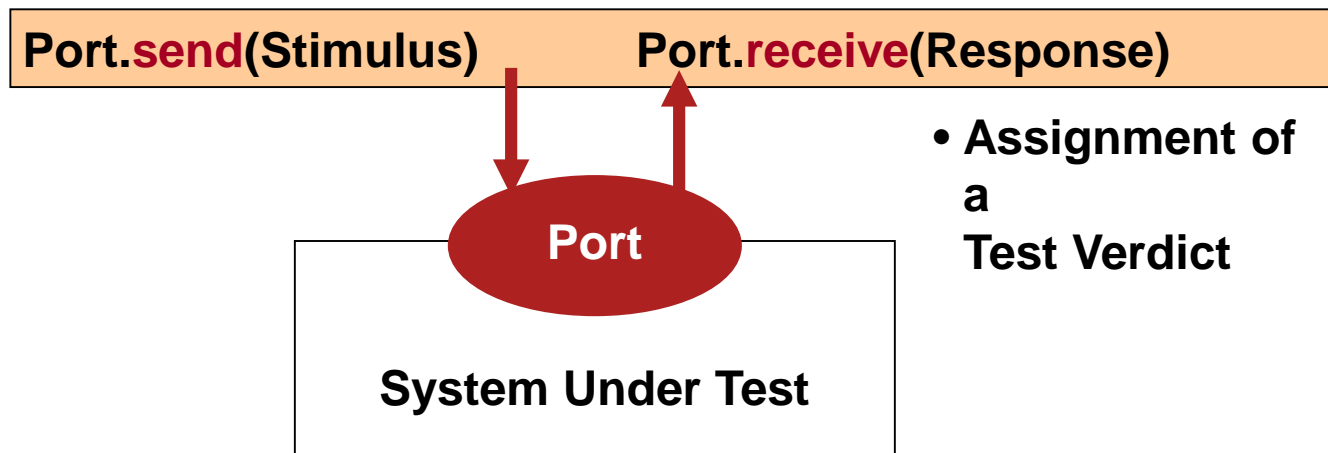


- **Used for *system and product* qualification and certification**

TTCN-3 IS DESIGNED FOR DYNAMIC TESTING (BLACK-BOX)

Abstract and platform-independent:

TTCN-3 Test Case



MAJOR LANGUAGE ELEMENTS OF TTCN-3 NOTATION

Compiled to C/C++ or Java:

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

IMPLEMENTATION USING TTCN-3

- **Type definitions:**

boolean, integer, float, bitstring, charstring, octetstring, hexstring, record, set, enumeration, union

- **Programming constructs:**

message: send/receive

procedure: call/getcall, reply/getreply, raise/catch

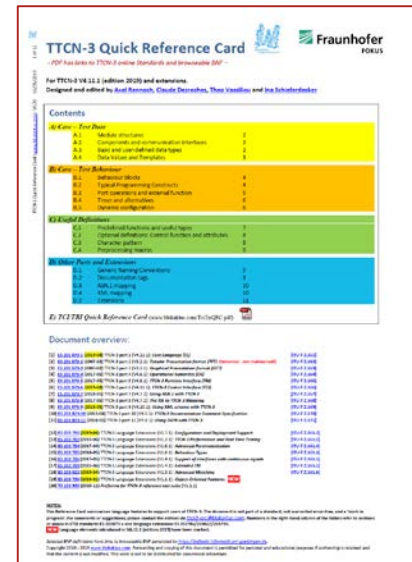
if-then-else, loops: for, while, do-while, functions, alternatives

component/port/timer control

- **Predefined functions:**

type conversion, lengthof (string), sizeof (records), ...

- **Overview:** e.g. TTCN-3 Quick Reference Card



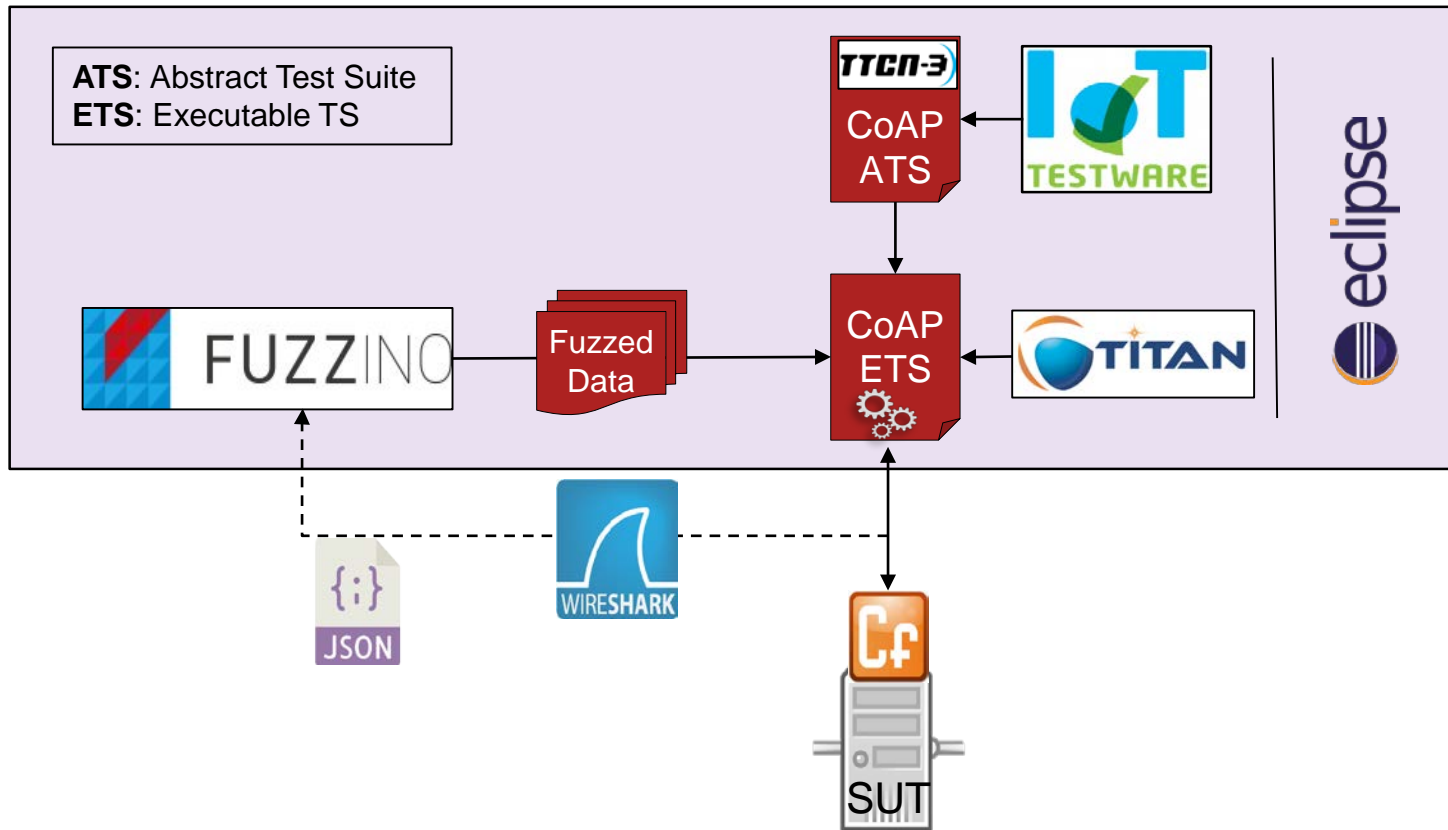
The image shows the cover and table of contents of the 'TTCN-3 Quick Reference Card'. The cover includes the title 'TTCN-3 Quick Reference Card', the subtitle 'TTCN has been a TTCN-3 online Standard and browserable PDF', and the logos for Fraunhofer FOKUS and the TTCN-3 Working Group. The table of contents lists various sections such as 'Contents', '1.1 Introduction', '1.2 Basic Concepts', '1.3 Data Types', '1.4 Control Structures', '1.5 Functions', '1.6 Components and Ports', '1.7 Timers', '1.8 Error Handling', '1.9 Test Cases', '1.10 Test Suites', '1.11 Test Results', '1.12 Test Logs', '1.13 Test Reports', '1.14 Test Summary', '1.15 Test Conclusion', '1.16 Test Appendix', '1.17 Test Bibliography', '1.18 Test Index', '1.19 Test Glossary', '1.20 Test Abbreviations', '1.21 Test Symbols', '1.22 Test Figures', '1.23 Test Tables', '1.24 Test Equations', '1.25 Test Formulas', '1.26 Test Diagrams', '1.27 Test Flowcharts', '1.28 Test Algorithms', '1.29 Test Procedures', '1.30 Test Methods', '1.31 Test Techniques', '1.32 Test Strategies', '1.33 Test Approaches', '1.34 Test Frameworks', '1.35 Test Environments', '1.36 Test Tools', '1.37 Test Languages', '1.38 Test Platforms', '1.39 Test Hardware', '1.40 Test Software', '1.41 Test Services', '1.42 Test Networks', '1.43 Test Systems', '1.44 Test Architectures', '1.45 Test Models', '1.46 Test Simulations', '1.47 Test Emulations', '1.48 Test Virtualization', '1.49 Test Clouds', '1.50 Test Edge Computing', '1.51 Test IoT', '1.52 Test AI', '1.53 Test ML', '1.54 Test DL', '1.55 Test Quantum Computing', '1.56 Test Blockchain', '1.57 Test Cybersecurity', '1.58 Test Privacy', '1.59 Test Ethics', '1.60 Test Law', '1.61 Test Policy', '1.62 Test Governance', '1.63 Test Standards', '1.64 Test Regulations', '1.65 Test Guidelines', '1.66 Test Best Practices', '1.67 Test Case Studies', '1.68 Test Examples', '1.69 Test Templates', '1.70 Test Patterns', '1.71 Test Recipes', '1.72 Test Hacks', '1.73 Test Tips', '1.74 Test Tricks', '1.75 Test Tricks of the Trade', '1.76 Test Secrets', '1.77 Test Mysteries', '1.78 Test Legends', '1.79 Test Myths', '1.80 Test Rumors', '1.81 Test Gossip', '1.82 Test Whispers', '1.83 Test Hearsay', '1.84 Test Rumor-Mongering', '1.85 Test Gossip-Mongering', '1.86 Test Whisper-Mongering', '1.87 Test Hearsay-Mongering', '1.88 Test Rumor-Mongering', '1.89 Test Gossip-Mongering', '1.90 Test Whisper-Mongering', '1.91 Test Hearsay-Mongering', '1.92 Test Rumor-Mongering', '1.93 Test Gossip-Mongering', '1.94 Test Whisper-Mongering', '1.95 Test Hearsay-Mongering', '1.96 Test Rumor-Mongering', '1.97 Test Gossip-Mongering', '1.98 Test Whisper-Mongering', '1.99 Test Hearsay-Mongering', '2.00 Test Rumor-Mongering'.

MQTT EVALUATION

Broker		PASS		FAIL		INCONC	
Name	Version	#	%	#	%	#	%
Mosquitto	1.5.5	90	85,71%	11	10,48%	4	3,81%
HiveMQ CE	2019.1	86	81,90%	15	14,29%	4	3,81%
lannister	v0.9.8	68	64,76%	33	31,43%	4	3,81%
Apache ActiveMQ	5.15.9	58	55,24%	43	40,95%	4	3,81%
Aedes	0.38.0	58	55,24%	43	40,95%	4	3,81%
RSMB	1.3.0.2	50	47,62%	51	48,57%	4	3,81%
Mosca	2.8.3	43	40,95%	58	55,24%	4	3,81%
Apache Apollo	1.7.1	34	32,38%	70	66,67%	1	0,95%

April 2019 – 105 Test Cases

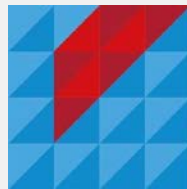
FUZZING APPROACH



FUZZINO RESULTS AND RESOURCES

Results for CoAP:

- Initially, 4421 fuzzed test data for CoAP were generated
- After sending the data to a (local) CoAP server, it crashed after date “1107”



FUZZINO

https://www.fokus.fraunhofer.de/de/sqc/security_testing

https://github.com/fraunhoferfokus/Fuzzino/blob/master/doc/Fuzzino_XML_Description.pdf

FUZZING DEMO

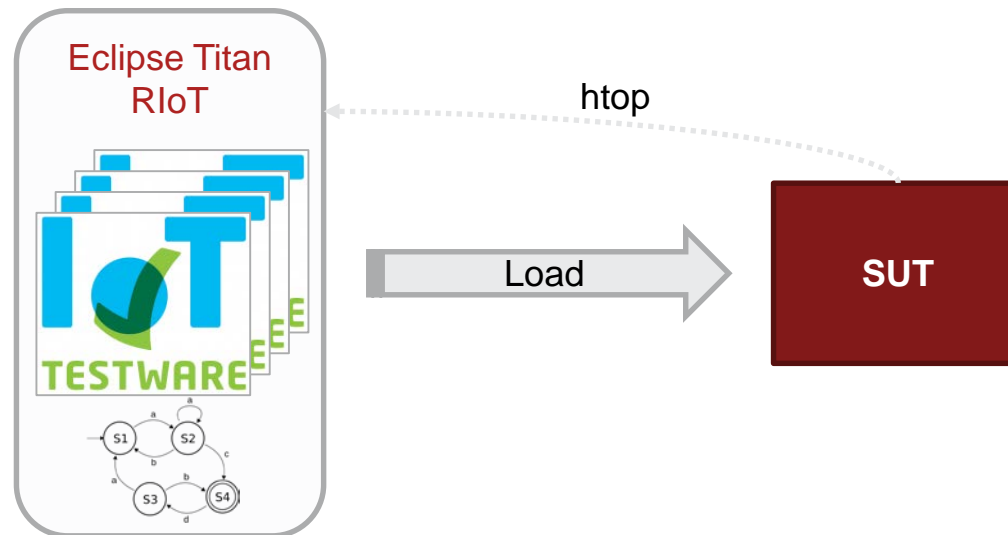
Detecting vulnerabilities using fuzzing

- Starting the SUT via SSH
- Starting the fuzzing campaign
- Detailed information gets collected in a CSV file
- **Waiting...->** Crash of SUT occurs
- Crash information gets collected in a CSV file

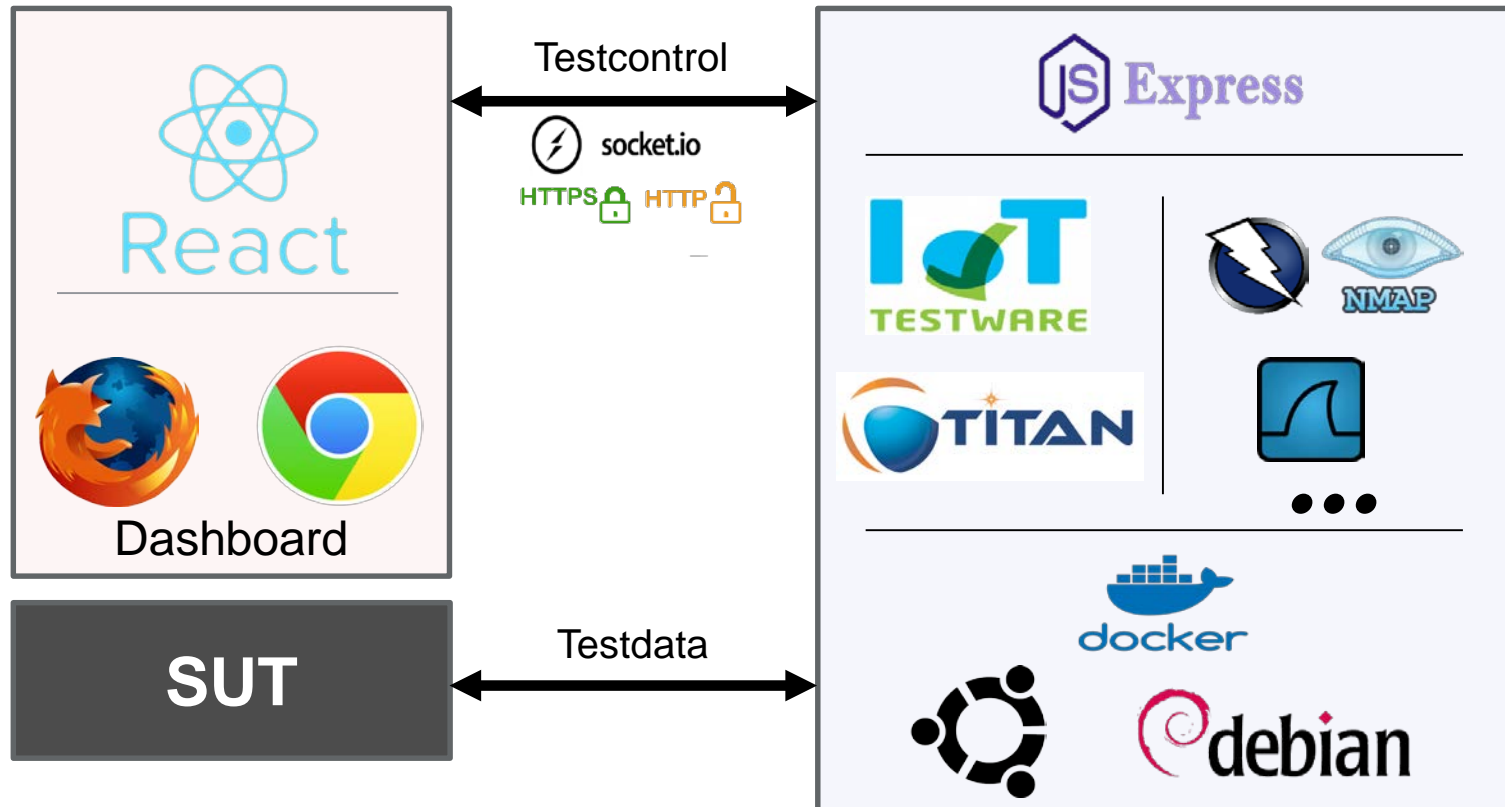
Performance increases when executed in console

- Malicious data gets produced
- **Waiting...->** Device gets rebooted after each crash
- **Waiting...** (for approximately 24h)... -> SUT crashed about 50 times
- Data on fuzzing and crash details gets merged
- A histogram reveals the error causing parts of the model
- The bug gets tracked down in a new test case
- The error causing PDU is found
- Packed in a narrow test case the error gets revealed

PERFORMANCE TESTING: HIGH-LEVEL VIEW



IOT-TESTWARE DASHBOARD



THE IOT-TESTWARE PROJECT











Standardization & Certification

- **New Working Group (TST)** will develop IoT test catalogues and specifications (not covered elsewhere)
- The **types of testing** include conformance, interoperability, security and performance testing
- The initial technical **focus** will be:
 - IoT **network layer**
(communication protocols, node connectivity, edge computing etc.),
 - **Basic security** of IoT devices

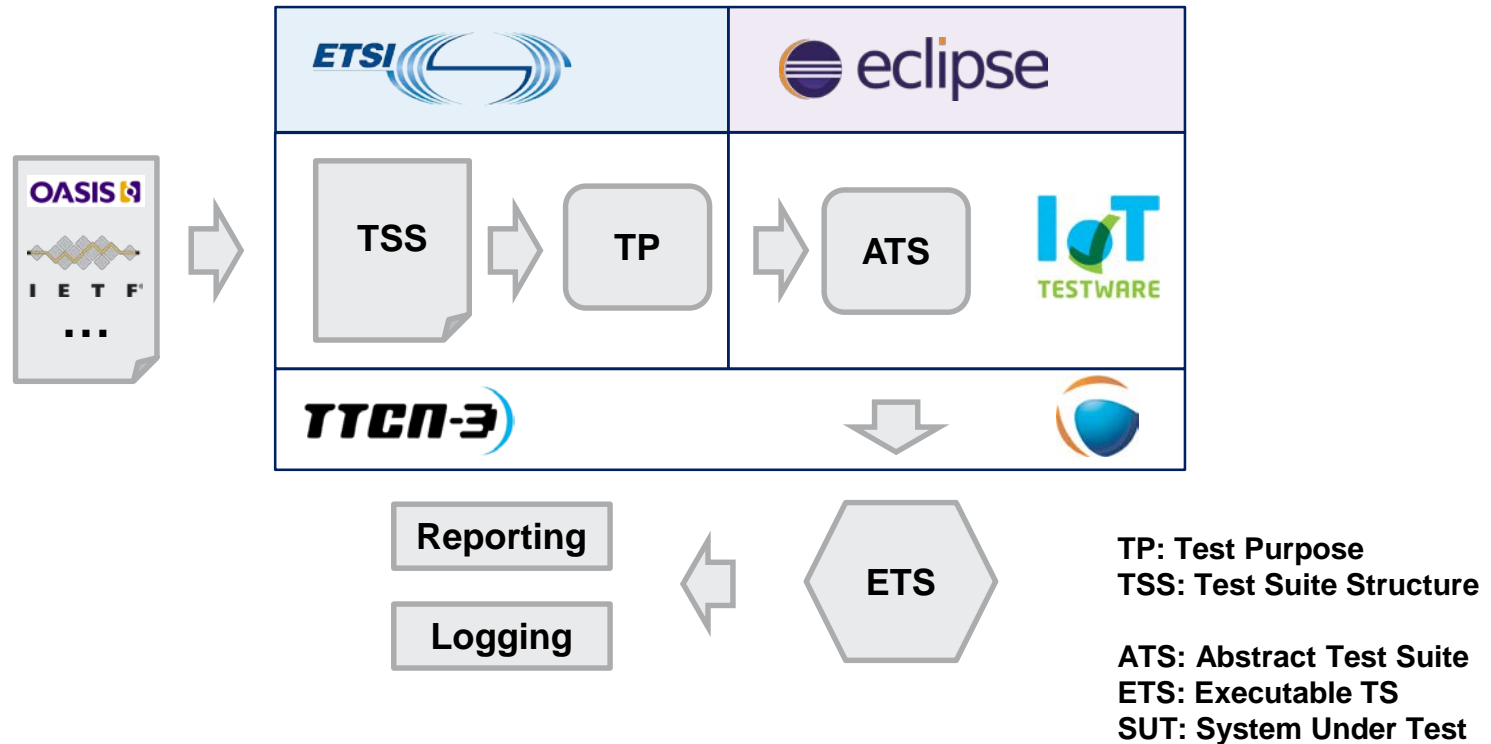


MTS TST WORK PROGRAMME

9 WIs, Work in progress, displaying 1 to 9				Displays	30
Work item number	Version	Current status	Next status	Rapporteur name	
MTS TST					
DTS/MTS-TST8 (TS 103 646)	 0.0.4	 Early draft (2019-05-21)	Stable draft	Wardaschka Andre	IEC 62443-4-2
DTS/MTS-TSTCoAP-1 (TS 103 596-1)	 0.0.4	 Early draft (2019-05-21)	Stable draft	Hackel Sascha	
DTS/MTS-TSTCoAP-2 (TS 103 596-2)		Start of work (2018-01-24)	Early draft	Hackel Sascha	
DTS/MTS-TSTCoAP-3 (TS 103 596-3)		Start of work (2018-01-24)	Early draft	Hackel Sascha	
DTS/MTS-TSTLoRaWAN (TS 103 598)	 0.0.1	Early draft (2019-03-27)	Stable draft	AHMAD Abbas	LoRaWAN
DTS/MTS-TSTMQTT-1 (TS 103 597-1)	 0.0.4	 Early draft (2019-05-21)	Stable draft	Pintar Bostjan	MQTT
DTS/MTS-TSTMQTT-2 (TS 103 597-2)		Start of work (2018-05-30)	Early draft	Pintar Bostjan	
DTS/MTS-TSTMQTT-3 (TS 103 597-3)		Start of work (2018-05-30)	Early draft	Pintar Bostjan	
DTR/MTS-TSTSecTM (TR 103 599)	 0.0.1	Early draft (2019-03-27)	Stable draft	AHMAD Abbas	Vul. database

<https://portal.etsi.org/tb.aspx?tbid=860&SubTB=860>

IOT-TESTWARE - BIG PICTURE



SUMMARY

- ✓ Advanced testing technology:
- ✓ Open source IoT-Testware (code):
- ✓ External (open source) SW
- ✓ Standardized IoT test purposes:



Thank you for your attention!

Ina.Schieferdecker@fokus.fraunhofer.de

Axel.Rennoch@fokus.fraunhofer.de

<https://www.fokus.fraunhofer.de/en/sqc>