Using TDL for Standardised Test Purpose Definitions

Philip Makedonski
Institute of Computer Science
University of Göttingen
Göttingen, Germany
makedonski@cs.uni-goettingen.de

Ilie-Daniel Gheorghe-Pop
SQC
Fraunhofer FOKUS
Berlin, Germany
ilie-daniel.gheorghe-pop@fokus.fraunhofer.de

Axel Rennoch
SQC
Fraunhofer FOKUS
Berlin, Germany
axel.rennoch@fokus.fraunhofer.de

Finn Kristoffersen
Cinderella ApS
Hvidovre, Denmark
finn@cinderella.dk

Boštjan Pintar
SINTESIO Foundation
Kranj, Slovenia
pintar@sintesio.org

Andreas Ulrich
Siemens AG
München, Germany
andreas.ulrich@siemens.com

***Abstract*—This article reports on experiences from the use of the ETSI Test Description Language (TDL) and its extension for structured test objective specification (TDL-TO) for the definition of functional and non-functional test purposes in the Internet of Things (IoT) domain. The experiences are based on results from different working groups at ETSI TC MTS and the ETSI Specialist Task Force (STF) 574, focusing on the definition of test purposes for functional, security, and performance testing of the CoAP and MQTT protocols as well as VxLTE interoperability testing.**

Keywords—Test description, test purposes, security, performance, interoperability, CoAP, MQTT, IoT, VxLTE

I. Introduction

The ETSI Test Description Language (TDL) [4] is a new domain-specific language for the specification of test descriptions and the presentation of test execution results developed and standardised at the European Telecommunications Standards Institute (ETSI) by the Technical Committee Methods for Testing and Specification (TC MTS). It consists of a standardised meta-model defining the relevant concepts, the relationships among them, and the associated semantics. Test descriptions in TDL are instances of the meta-model. Concrete representations of the instances can take the form of structured text, graphical shapes, and machine-readable assets for model exchange between tools.

While the main focus of TDL is on the specification of elaborated test descriptions, including relevant test data, test configurations, and test scenarios, standardised extensions provide means to widen the scope of TDL to accommodate certain use cases. The Structured Test Objective (TDL-TO) extension [1] adopts established concepts for the specification of test purposes, based in part on experiences with the previously applied Test Purpose Language (TPLan) [5]. TDL-TO integrates these concepts in TDL models by providing a (semi-) structured means for expressing test objectives and mapping the resulting structures to TDL model elements, which can then be used to streamline the transition from test purposes to test descriptions.

Makedonski et al. [9] reported on a study initiated by the ETSI Centre for Testing and Interoperability (CTI) to assess the applicability of TDL for the standardisation of diverse technologies. It resulted in a set of non-trivial examples of test purposes and test descriptions serving as a starting point for a more widespread use of TDL within ETSI standardisation groups. This article discusses more recent experiences from the use of TDL and in particular TDL-TO for Internet of Things (IoT) test purpose definitions.

At ETSI, the TST working group within TC MTS develops studies, guidelines, test catalogues, and test specifications for specific ICT technologies that are not already covered by existing ETSI Technical Bodies (TBs). The initial technical focus of TST is on:

- IoT network communication protocols, node connectivity, edge computing,
- IoT data accumulation and aggregation, and
- IoT application interfaces, business processes.

This article presents the current work within TST and the achievements regarding the application of TDL-TO for the IoT domain.

Section II provides the motivation and scope for TDL, in particular the ideas and concepts behind TDL-TO and corresponding tool support. Section III presents results from using TDL-TO for the IoT protocols CoAP and MQTT. It addresses functional, security, and performance testing aspects. Further experiences from the test specification project defining interoperability tests for Voice/Video over LTE (VxLTE) are discussed as well. Finally, a short conclusion and an outlook are provided in Section IV.

II. TDL

A. Motivation for TDL

The growing scale, complexity, and interconnectedness of systems, compounded by the need for faster time to market and faster iteration cycles, increases the need for more scalable and automatable approaches for the development of test specifications. The ETSI test development process in ETSI TR 102 840 [7], based on the joint ISO/IEC and ITU-T OSI conformance

978-1-7281-8915-4/20/$31.00 ©2020 IEEE
DOI 10.1109/QRS-C51114.2020.00091

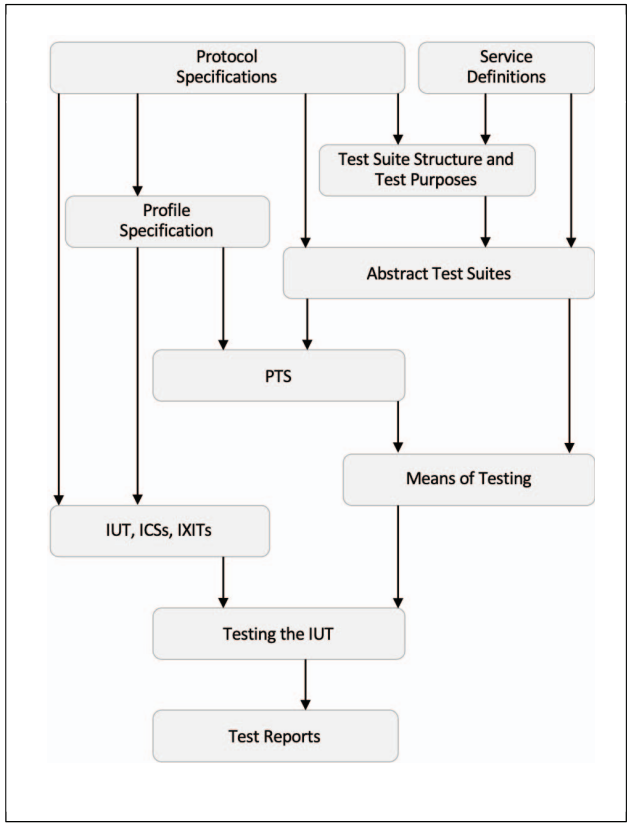

Fig. 1. Conformance testing methodology (derived from ITU-T X.290)

testing methodology and framework illustrated in Fig. 1 [9], outlines a sequence of steps for the derivation of executable test cases from a base standard. The intermediate artefacts at the different levels of abstraction are intended for specific audiences, such as standardisation experts, technology experts, and test engineers. Major time-consuming steps involve the development of test purposes and the implementation of the abstract test suites. Additional input in such processes may be retrieved from profile test specifications (PTS), implementation conformance statements (ICSs), and implementation extra information for testing (IXITs). In protocol testing, the latter are mapped to PICSs and PIXITs, correspondingly. Traditionally, many of the steps have been document-oriented and lack sufficient tooling and automation support to enable improved quality and better scalability of the test specification and documentation to meet the accelerating development processes.

At the level of executable tests, ETSI TC MTS developed and is continuously maintaining the Testing and Test Control Notation version 3 (TTCN-3), published as ETSI ES 201 873-1 [6], over the past 20 years. At the other end of the spectrum, the Test Purpose Language (TPLan) [5] laid some groundwork for the structured specification of test purposes, indicating what needs to be tested in a declarative manner. Before TDL came along, there was a methodology gap between the high-level expression of what needs to be tested expressed in test purposes described in TPLan or simple loosely structured prose, and the complex coding of the executable tests in TTCN-3. TDL fills that gap. TDL adopted a more modern model-based approach to domain-specific language development, capitalising on the benefits provided by mature modelling frameworks and technologies. This paved the way for a more streamlined approach to test specification development at a faster pace and a greater scale.

B. Scope of TDL

The TDL meta-model (TDL-MM) (part 1 of [4]), provides a common ground for customised representations by means of different concrete syntaxes. The standardised graphical representation (TDL-GR) (part 2 of [4]) defines corresponding shapes for the representation of model elements in TDL specifications. It can also serve as a blueprint for customised graphical representations tailored to specific domains and stakeholders. The TDL exchange format (TDL-XF) (part 3 of [4]) enables the interchange of TDL models between tools.

Complementary to the core constituents of TDL, extensions provide additional capabilities to accommodate specific application scenarios. The Structured Test Objective (TDL-TO) extension (part 4 of [4]) integrates concepts related to the specification of test purposes in TDL models. The Extended Test Configurations (TDL-TC) (part 7 of [4]) extension provides means for a more sophisticated management and reuse of test configurations.

Finally, mappings to other languages provide bridges to other technologies in a standardised manner, to streamline the transition to executable tests, for example. The mapping to TTCN-3 (TDL-T3) (part 6 of [4]) provides a standardised mapping to TTCN-3, which can also serve as a blueprint for the mapping to other target execution platforms. The UML profile for TDL (UP4TDL) (part 5 of [4]) instead provides a way to use TDL within UML based environments.

Recent maintenance and evolution efforts added support for inheritance in data and configuration related TDL elements, enabling better reuse in TDL. Future endeavours may bring standardised refinement of test purposes into test descriptions, as well as standardised mappings to other execution platforms.

C. TDL-TO

Without the TDL-TO extension, TDL provides simplified and generic means for the specification of test objectives by means of informal text. The TDL-TO extension introduces additional concepts to support the specification of (semi-) structured test objectives in a more formalised manner as well as a concrete syntax notation for representing the additional concepts. The additional concepts are related to the specification of the domain, including events and entities, as well as the structure of the test objectives, defining event occurrence sequences for initial conditions, expected behaviour, and final conditions. The notation for the representation of structured test objectives is graphical in nature. However, the contents of the event occurrences are in the form of an embedded structured language that is close to natural English, where entity and event references are surrounded by keywords and descriptive text. This approach provides some formalisation to the specification of test objectives in order to facilitate the validation of test objectives and paves the way for transforming them to test descriptions or test case skeletons, while still retaining a natural language feel. In

```
Domain {
    pics:
    - ACCEPTS_REQUESTS
    - AUTHENTICATES_REQUESTS
    ;
    entities:
    - client
    - server
    - gateway
    - authenticator
    ;
    events:
    - sends
    - receives
    - registers
    - acknowledges
    - logs
    - initiates
    - confirms
    ;
}
```

Fig. 2. Domain specification with TDL-TO

addition to the standardised graphical representation, a purely textual notation is provided as an informative example.

The domain specification concepts add the notions of abstract events and entities. Reusable event occurrence templates help in defining fixed patterns for event occurrences where individual parts can be overridden when the template is used within a structured test objective. Finally, Protocol Implementation Conformance Statements (PICS) provide an indication in which implementations a particular test objective is applicable. Fig. 2 showcases an example with domain specification concepts.

Structured test objectives contain additional descriptive meta-information, including an optional reference to a related test configuration, as well as a selection of applicable PICS. The initial conditions, expected behaviour, and final conditions are comprised of event occurrence sequences, which make up the major contribution of the TDL-TO extension. Fig. 3 showcases an illustrative example for a minimal test purpose specified with TDL-TO. The main part of the test purpose is contained in the expected behaviour block. It is comprised of one or, more commonly as in the example, two blocks, describing the basic stimulus-response pattern. *When* a sequence of one or more events occurs as a stimulus, *then* a sequence of one or more events occurs as a response to the first sequence. In the example, each sequence is comprised of a single event occurrence. Each event occurrence references one or, for more completeness, two (as in the example) entities that are involved in the event occurrence. Event occurrences also include references to the events that occur. In this example, the events are *sends* and *receives*, but they can also be arbitrary events or states defined in the domain specification, such as *moves*, *is*, *has*, etc. An event argument provides additional information regarding the event occurrence, usually in the form of a data specification, it may also indicate a defined state or desired position. Finally, time labels *(.)* and time constraints *(!)* can be used to express temporal dependencies between event occurrences and corresponding temporal requirements.

The initial and final conditions are omitted in the example for brevity. They follow a similar structure as the expected behaviour, but only include one block with the relevant sequences of event occurrences which define the state of the System Under Test (SUT) before and after the test, contained in a corresponding *with* sequence of one or more event occurrences within the initial and/or final conditions blocks.

The textual representation is convenient for editing and versioning. The graphical representation is more convenient for embedding in documents. The corresponding graphical representation for the example in Fig. 3 discussed above is shown in Fig. 4. As part of the graphical layout, some keywords, such as *entity*, as well as special characters may be omitted for better readability. Similarly, optional compartments which are empty may be omitted in the final graphical representation.

The structured test objectives can then be refined into more detailed test descriptions, either manually or (semi-) automatically (e.g. by means of model transformations). A graphical representation of the derived test description is shown in Fig. 5. It can be used for documentation and communication purposes, or as the basis for further refinement, including transformation to

```
Test Purpose {
    TP Id GET_RESOURCE
    Test objective "A resource shall be received within 5ms after a request"
    PICS Selection ACCEPTS_REQUESTS
    Expected behaviour
    ensure that {
        when {
            (.) at time point t1:
                the client entity sends a request
                        containing uri set to "/resource/";
                    to the server entity
        }
        then {
            (!) within 5ms after t1:
                the client entity receives a response
                        containing body set to "resource: {...}";
                    from the server entity
        }
    }
}
```

Fig. 3. Test purpose example with TDL-TO (textual representation)

TP Id	GET_RESOURCE
Test Objective	A resource shall be received within 5ms after a request
PICS Selection	ACCEPTS_REQUESTS
Expected Behaviour	

```
ensure that {
    when {
        . at time point t1:
            the client sends a request
                    containing uri set to "/resource/"
                to the server entity
    }
    then {
        ! 5ms after t1:
            the client receives a response
                    containing body set to "resource: {...}"
                from the server entity
    }
}
```

Fig. 4. Test purpose example with TDL-TO (graphical representation of Figure 3)

executable test cases, e.g. specified in TTCN-3 with the help of the standardised mapping of TDL to TTCN-3 (part 6 of [4]), as shown in Fig. 6.

D. Tool support for TDL

The TDL Open Source Project (TOP) [22] fosters the shared development and testing of contributed tools to manage TDL models and related assets, such as generated documentation, and different representations of the models. The TOP serves as a common starting point to accelerate the adoption of TDL and lowers the barrier of entry for both, users and tool-vendors. In addition, it offers a platform to validate the TDL standards and check their applicability, consistency, and usability.

The TOP is built on top of the Eclipse platform [23] and associated technologies, including the Eclipse Modelling Framework (EMF) [24], Xtext for textual editor implementation [25], and Sirius for graphical editor implementation [26].

For the realisation of the graphical representation of structured test objectives, it was determined that they need to be exported as tables in a Word document to better serve the end-users. Specific templates are provided to accommodate established notations and guidelines in the different working groups. While the original implementation for the Word generation was based on the docx4j library [14], recently the Apache POI library [15] was adopted instead.

Traditionally, the TOP follows the development of the TDL standards. While it is used to prototype and explore different solutions for new features, users typically only gain access to them after the standards are published. To involve users more actively in the maintenance and evolution of TDL and also gain early feedback on potential solutions, a pilot implementation-first approach for a planned feature is pursued with the *variants* specification. The *variants* specification provides means for the reuse of whole test purposes, where variants of the test purpose with different combinations of meta-information and event arguments can be specified without the need to duplicate all the other parts that remain unchanged. This approach does present some challenges in the absence of a standardised description of the feature that users can rely on. However, it also provides an opportunity for users to be involved in determining how the feature should work. This way, users can try early on if a proposed solution fulfils their requirements, rather than waiting for another maintenance cycle to address new or changing requirements. If this new approach proves to be successful, it may be taken up for other features in the future as well, in order to pursue an agile and user-driven evolution of TDL.

III. Experiences with TDL

Test purpose definitions have been developed and used in industry and research, in and outside of standardization for many years. In practice, we have seen many templates for such test purpose definitions for various public test suites [3]. The minimal parts cover an identifier, status information (e.g. mandatory), references (for the related requirements) and some (natural language) text field for a textual description of the test purpose, as illustrated in the example in Fig. 7. More detailed test purpose descriptions may also cover e.g. selection criteria (PICSs) and initial conditions, and distinguish between a textual summary, the "core" definition of the condition/expectation of the test, and some optional comments/notes. Furthermore, advanced test purposes include additional information for dedicated parameters. A higher level of detail and supplementary information about the test purpose enable an easier and accurate implementation of the test case within a chosen test framework.

Historically, we can observe a large variety in the templates being used for test purposes as well as in corresponding levels of detail and supplementary information supplied with the test purposes. Larger test development projects have to decide about the chosen test purpose description style and conventions, often leading to yet more templates and differences in the information provided as part of the test purpose specifications.

Following the advanced test methodology developed within ETSI MTS, it was decided to apply TDL-TO for the definition of test purposes within the work on the IoT-Testware [2] project.

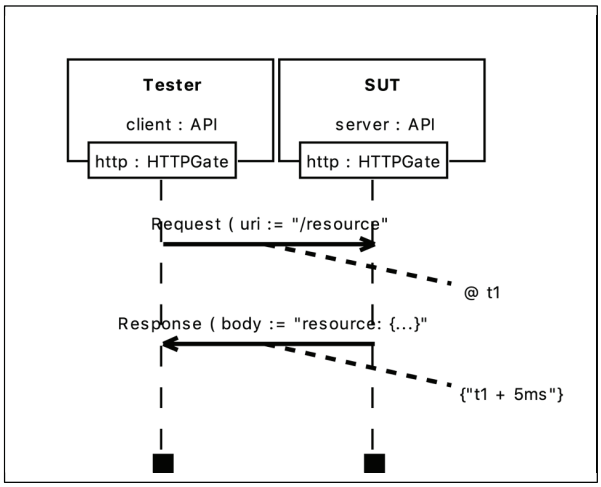

Fig. 5. Test description in TDL-GR derived from Fig. 3

```
function setupTestConfiguration_ClientServer()
    runs on MTC_BasicClientServer {
  client := API.create;
  map (client : http_to_server_http ,
       system : server_http);
}

function TD_GET_RESOURCE()
    runs on MTC_ClientServer {
  client.start(TD_GET_RESOURCE_client_main());
}

function TD_GET_RESOURCE_client_main()
    runs on API {
  http_to_server_http.send(Request : {
    uri := "/resource"
  });
  http_to_server_http.receive(Response : {
    body := "resource: {...}"
  });
}

testcase tc_TD_GET_RESOURCE ( )
    runs on MTC_ClientServer
    system SYSTEM_ClientServer {
  setupTestConfiguration_ClientServer() ;
  TD_GET_RESOURCE() ;
  all component.done ;
}
```

Fig. 6. TTCN-3 test code derived from Fig. 3 (based on part 6 of [4])

Based on our experiences, we have determined that it is essential to define test scenarios in a formalised way to avoid misinterpretation and to enable the application of suitable tools supporting consistency checking and formatting during development and maintenance.

Major arguments for the application of TDL-TO in the IoT Testware project are the possibility of expressing informal test specifications, the support of simple description structures (event occurrence sequences), global, domain-specific keyword definitions and the focus on a single test observation for a pass/fail verdict criterion.

TPId:	**SIP_RG_RT_V_009**
Status:	Mandatory
Ref:	RFC 3261 section 10.2.
Purpose:	Ensure that the IUT, in order to be registered, sends a REGISTER request to its registrar, with the same URI in the From and the To header.

Fig. 7. Simple test purpose for the SIP protocol

In the following sections we present the specific experiences obtained from the IoT Testware project for different testing types and from an ETSI interoperability test specification project on Voice and Video services over Long Term Evolution (LTE) networks (VxLTE).

A. *Functional Tests*

From our experience, most standardised test purpose catalogues developed for conformance and interoperability tests focus mostly on the telecommunications domain. Other domains, such as industrial production or IoT, are generally underrepresented. In the context of a national research project in Germany [13] the focus has been on the IoT protocols CoAP and MQTT.

Each test purpose (TP) has been written in TDL-TO and thus in a structured manner which is consistent with all other TPs. The intention was to formalise the TPs and provide a user-readable format by generating tables out of the TDL-TO specification.

As a result of our project, two ETSI technical specifications have been produced covering conformance test purposes: 164 tests for CoAP and 162 tests for MQTT. The most frequently used language elements for TDL-TO were conditions, domains (*entities*, *events*, *pics*), and event sequences.

B. *Security Tests*

The work regarding security testing addresses general IoT security considerations [11] and specific test purposes and guidelines about selected IoT protocols [16], [17]. The collective ideas presented in the documents are enriched with example test purposes to outline possible test case implementations and attack scenarios.

Starting from technical requirements for Industrial Automation and Control System (IACS) components, first a selection of basic requirements was created to build a profile for a base security level. A total of 43 test purposes were defined using TDL-TO and included in ETSI TS 103 646, which is currently under review among ETSI members. During this work, it became clear that the application of TDL-TO emphasises the test developer perspective. In particular, it requires the clear identification of the major verdict criteria for the pass/fail decision. The viewpoint is shifted from the system design and implementation approach to the logic of conditions, triggers, and expected outcomes to validate the required system security.

The technical scope of the defined test purposes for the IACS components contains several important security aspects, among others:

TP Id	TP_CR_1_1_CVE_2018_14367_1
Test Objective	Ensure that the IUT does not crash by using an invalid code in the CoAP header
Reference	CVE-2018-14367

Initial Conditions

```
with {
    the IUT being_in the initial_state
}
```

Expected Behaviour

```
ensure that {
    when {
        the IUT receives a request message containing
            version indicating value 1,
            msg_type indicating value 0, //Confirmable
            token_length indicating value 0,
            code indicating value NULL, //broken message
            msg_id corresponding to MSG_ID1
    }
    then {
        the IUT is pingable
    }
}
```

Fig. 8. CoAP security test purpose example

- Identification and authentication
- Account changeability
- Strength of password
- Session lock
- Input validation
- Information confidentiality
- Use of cryptography
- Update support

From the language point of view, the TP specification used domain definitions (*entities*, *events*, *pics*), event sequences time labels, and time constraints.

Additional work regarding test purpose definitions for security testing has been done for dedicated IoT protocols. The technical specifications address CoAP and MQTT security aspects. They provide general aspects about security testing of the IoT protocols, e.g., security testing techniques (fuzz testing, penetration testing, spoofing, amplification etc.), test configurations, but also protocol specific test purposes for security. A major part addresses test purposes focusing on testing for vulnerabilities, known and published by CVE, e.g. CVE_2018_14367 [12], as illustrated in Fig. 8.

The TOP toolset was well-suited for the task. However, in the version of the TOP tools used for this project the automatic translation to the Word tabular format lost some existing syntax formatting (e.g. bold style for keywords) and missed the listing of the catalogue of defined keywords in the common domain definitions. A change request to improve these features has been reported to the TOP project as the improvements will benefit other users as well. The TOP maintenance team is devoted to adding addressing this change request as well as adding other improvements.

C. Performance Tests

For communications protocols in general, specifically for protocols working in constrained environments, performance requirements typically target transmission volume, robustness, and latency. In the case of IoT protocols, testers must be able to specify the test type, duration, load intervals, as well as performance related thresholds or constraints.

Performance tests provide a basis for benchmark testing and performance evaluation of protocols. Performance test purpose descriptions heavily use the timing expressions of TDL-TO. In addition to the domain specifications such as *events*, *pics* and *entities*, they allow for describing performance benchmarks of IoT protocols such as CoAP and MQTT using multiple types of tests. For these protocols, current work includes multiple performance aspect tests, including:

- Load tests
- Stress tests
- Endurance tests

TDL-TO provides the elements and syntax to be able to write performance test purposes, as illustrated in Fig. 9, depicting a test purpose for a stress test using pings to evaluate the server response time during load spikes. It is actively used for standardization work for MQTT and CoAP protocols at ETSI [18][19]. Further details on the performance benchmarking methodology, including additional samples using TDL-TO, can be obtained from [27].

D. Interoperability test specification

In the ETSI test suite specification project for voice and video services over LTE [20], TDL-TO was used to specify the conformance criteria utilised in interoperability test descriptions. The TDL-TO test purposes were defined for all interfaces of the identified test configurations used in interoperability testing.

TP Id	Ping_Stress_Test
Test Objective	Determine if the IUT (server) can handle the given spiking load for a determined period of time without exceeding the delay threshold within a given acceptable message loss rate.
PICS Selection	PERFORMANCE

Expected Behaviour

```
ensure that {
    when {
        . at time point t1:
            the clients sends multiple Ping messages and assures the RATE and
        ! during INTERVAL after t1:
            the IUT receives several Ping messages and
        . at time point t2:
            the clients sends multiple Ping messages and assures the SPIKE_RATE and
        ! during INTERVAL after t1 :
            the IUT sends multiple messages containing
                code indicating value 7.03
    }
    then {
        the IUT assures the response messages and
        the IUT assures the packet_loss_limit and
        the IUT assures the DELAY
    }
}
```

Fig. 9. Performance test purpose example with TDL-TO

```
Configuration {
  Interface Type defaultGT accepts DiameterMessage;
  Component Type DiameterComp with
    gate g of type defaultGT
  ;
  Test Configuration CF_VxLTE_INT containing
    SUT component EPC_PGW_A of type DiameterComp
    SUT component EPC_PCRF_A of type DiameterComp
    SUT component S_CSCF_A of type DiameterComp
    SUT component I_CSCF_A of type DiameterComp
    SUT component P_CSCF_A of type DiameterComp
    SUT component HSS_A of type DiameterComp
    SUT component EPC_MME_A of type DiameterComp
    SUT component IMS_AS_A of type DiameterComp
    connection between EPC_MME_A.g and HSS_A.g
    connection between EPC_PGW_A.g and EPC_PCRF_A.g
    connection between EPC_PCRF_A.g and P_CSCF_A.g
    connection between HSS_A.g and S_CSCF_A.g
    connection between HSS_A.g and I_CSCF_A.g
    connection between IMS_AS_A.g and HSS_A.g
  ;
  Test Configuration CF_VxLTE_RMI containing
    SUT component EPC_PGW_B of type DiameterComp
    SUT component EPC_PCRF_A of type DiameterComp
    SUT component EPC_PCRF_B of type DiameterComp
    SUT component P_CSCF_B of type DiameterComp
    SUT component HSS_A of type DiameterComp
    SUT component EPC_MME_B of type DiameterComp
    connection between EPC_MME_B.g and HSS_A.g
    connection between EPC_PGW_B.g and EPC_PCRF_B.g
    connection between EPC_PCRF_A.g and EPC_PCRF_B.g
    connection between EPC_PCRF_B.g and P_CSCF_B.g
  ;
} // End of Configuration section
```

Fig. 10. TDL-TO test configuration example

Packages with common domain definitions (event, entities, configurations, and data) were specified and imported in the packages containing the test purposes for the individual interfaces. Fig. 10 illustrates the specification of two test configurations, CF_VxLTE_INT and CF_VxLTE_RMI. The test configurations define the components, their role (tester or SUT), and which communication may take place between the components over explicitly defined connections. Tester components have not been specified since with interoperability testing user equipment is planned to be used manually to execute each and every test scenario in the first phase. Over the complete signalling path, monitoring points are created to observe the behaviour on each interface along the complete signalling path. Therefore, only SUT components were defined along their ports to identify the connections between them.

The test purpose definition in TDL-TO follows the overall approach illustrated in Fig. 1. In addition to the information retrieved from the requirements of the base protocol standard, we also used information from an abstract test suite specification related to an earlier release of the base protocol, effectively applying a re-engineering approach. For requirements that were still valid, the test purposes from the previous test suite specified in TPLan were translated into TDL-TO using the textual syntax supported by TOP. In addition, TDL-TO test purposes were specified for new requirements of the base protocol specification. The TDL-TO test purposes were converted to the tabular representation format in Word using TOP.

In the TDL-TO behaviour specifications of the test purposes, the event data instances were defined using information from the existing abstract test suite data definition libraries. This way, the transformation from test purposes to the TTCN-3 test case implementation was also supported. More general guidelines on transformation of TDL towards executable tests are presented in the TDL Reference Implementation technical report [21].

In this project, more than 300 TDL-TO test purposes were specified covering the functional requirements of the base standard. These test purposes were used for the definition of interoperability test descriptions and the development of the abstract test suite specification in TTCN-3. Using TDL-TO and TOP helped the test specification process to ensure syntactically correct and semantically consistent test purposes. In addition, the

early formalization of data in TDL-TO was useful for the transformation to the interoperability test descriptions and abstract test suite.

IV. Conclusions and Future work

TDL has been applied in several standardization projects and identified as a suitable language for the definition of test purposes covering different testing types including conformance, security, performance, and interoperability. The most relevant part of TDL in this work was the TDL-TO extension, which was designed for the definition of test purposes.

It is expected that TDL will be used more and more in international test specification projects. ETSI continues to maintain TDL, with a forward-looking roadmap for the evolution of the language to improve its usability and widen its user base.

Acknowledgment

The work is supported by the experts from the ETSI TC MTS working group TDL and TST. The work on TDL has been partially funded by the European Telecommunications Standards Institute (ETSI) in the context of the Specialist Task Force (STF) projects 454, 476, 492, 522, and 577 between 2013 and 2020. The work on VxLTE test specification has been performed by the experts in the STF 574 project, funded by the ETSI. The work on IoT test purposes has been partly funded by the German Federal Ministry of Economics and Technology with its project IoT-T.

References

[1] ETSI ES 203 119-4, Methods for Testing and Specification (MTS); The Test Description Language (TDL); Part 4: Structured Test Objective Specification (Extension)

[2] A. Kaiser and S. Hackel: “Standards-Based IoT Testing with Open-Source Test Equipment,” IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C), June 2019. (DOI 10.1109/QRS-C.2019.00085)

[3] ETSI catalog of standardised public test suites, http://www.ttcn-3.org/index.php/downloads/publicts/publicts-etsi

[4] ETSI Test Description Language (TDL) standards (ES 203 119), https://tdl.etsi.org/index.php/downloads

[5] ETSI TPLan: A notation for expressing Test Purposes, https://portal.etsi.org/Services/Centre-for-Testing-Interoperability/ETSI-Approach/Testing-Languages/TPLAN

[6] ETSI TTCN-3 standards (ES 201 873), http://www.ttcn-3.org/index.php/downloads/standards

[7] ETSI TR 102 840: Model-based testing in standardisation, https://www.etsi.org/deliver/etsi_tr/102800_102899/102840/01.02.01_60/tr_102840v010201p.pdf

[8] IEC 62443-4-2: Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components, https://webstore.iec.ch/publication/34421

[9] ITU-T Recommendation X.290, OSI conformance testing methodology and framework for protocol - general concepts.

[10] P. Makedonski et al.: Test descriptions with ETSI TDL, https://link.springer.com/article/10.1007/s11219-018-9423-9

[11] ETSI: MTS Test specification for foundational Security IoT-Profile, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=54751

[12] Common Vulnerabilities and Exposures, https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-14367

[13] IoT-T project, https://www.iot-t.de/en/

[14] Docx4j library: https://www.docx4java.org/trac/docx4j

[15] Apache POI - the Java API for Microsoft Documents, https://poi.apache.org/

[16] ETSI Test Specification for CoAP; Part 2: Security Tests, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=54409

[17] ETSI Test Specification for MQTT; Part 2: Security Tests, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=54410

[18] ETSI Test Specification for CoAP; Part 3: Performance Tests, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?wki_id=54412

[19] ETSI Test Specification for MQTT; Part 3: Performance Tests, https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?wki_id=54411

[20] ETSI TS 103 653-1, Core Network and Interoperability Testing (INT); VoLTE/ViLTE interoperability test description over 4G/early 5G in physical/virtual environments; Part 1: Test Purposes & PICS for VoLTE/ViLTE interoperability;

[21] ETSI TR 103 119, Methods for Testing and Specification (MTS); The Test Description Language (TDL); Reference Implementation

[22] ETSI TDL Open Source Project (TOP), https://tdl.etsi.org/index.php/open-source

[23] Eclipse Platform, https://projects.eclipse.org/projects/eclipse.platform

[24] Eclipse Modeoing Framework (EMF), https://www.eclipse.org/modeling/emf/

[25] Xtext, https://www.eclipse.org/Xtext/

[26] Sirius, https://www.eclipse.org/sirius/

[27] I.D. Gheorghe-Pop et al.: “A Performance Benchmarking Methodology for MQTT Broker Implementations”, IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2020.