Towards formalized test specifications for IEC 61850

Christof Brandauer, Georg Panholzer
Salzburg Research
{firstname.lastname}@salzburgresearch.at
Salzburg, Austria

Stephan Pietsch
Testing Technologies IST GmbH
pietsch@testingtech.com
Berlin, Germany

Abstract

This paper is concerned with testing in the context of IEC 61850. Currently, a paper and pen method is used to create prose descriptions of test cases in a table-based template. The test cases, while still remaining on an abstract level, could be significantly enhanced by adding more detail and making use of a (more) formal specification language. To this end we have studied other approaches to (conformance) testing, in particular TTCN-3 which is an open, standardized testing technology. In the paper it is demonstrated how existing UCAIug client tests are mapped to abstract TTCN-3 test cases. With the addition of SUT-specific adapters these test cases are then executed against a commercial IEC 61850 client implementation. Finally, we contrast this approach with an UML-based test development methodology.

1. Introduction

The inter-dependent processes of data acquisition, communication and (often automated) control of the smart grid are highly complex and stringent requirements in terms of correctness, standard conformance, interoperability, performance, and security are imposed on the components involved. Testing must play a major role in meeting these requirements – not least because the electric grid is a (if not the topmost) critical infrastructure.

Our studies on the current state of testing are focused on the multi-part standard IEC 61850. Its development started in 1994 with the goal of developing a comprehensive world-wide standard for the design and operation of substation automation. The main requirements were to advance beyond a growing set of proprietary, incompatible and non-comprehensive approaches to communication solutions in substation automation and to define a global standard that facilitates interoperability and integration. While the initial focus was on the electric substation, the standard has since been extended in several directions and is nowadays employed more broadly for general power utility automation tasks.

Part 10 of IEC 61850 is dedicated to conformance and performance testing. A conformance test has to include documentation and version control (IEC 61850-4), configuration (IEC 61850-6), data model (IEC 61850-7-3 and -7-4) and mapping of ACSI models and services (IEC 61850-7-2). Moreover, the testing "ecosystem" is explained including how to certify a tester. Part 10 identifies the areas to be tested (e.g., association, reporting, etc.) and introduces 167 server test cases for the ACSI mapping, all of which are mandatory if supported by the system under test (SUT). The positive and negative test cases are briefly described. Based on this framework, the testing subgroup of the Utility Communication Architecture International user group (UCAIug) elaborated more detailed test procedures for servers as well as clients, performance of fast event distribution with Generic Object Oriented Substation Events (GOOSE) and extended server reports.

As a common denominator, all these test cases are described in prose using the table template as required by IEC 61850-10. Figure 1 shows a simple client association test case.

Such brief prose descriptions naturally leave a lot of room for interpretation and many aspects of the test, some of which have certainly been in mind by the test developers, are not present in the test cases. As an example, it is not specified how to verify that the association has indeed been established. Where are the Points of Observation and Control? What exactly are the criteria against which the expected Associate.response+ reply has to be matched? Which protocol fields are required? Do some of them need to contain specific values? Which are irrelevant for the given test? It is obvious that the test cases, while still remaining on the abstract level, could be significantly enhanced and formalized. We have thus studied other approaches to conformance testing with a special focus on communication protocols.

cAss1	Associate and force client to release a TPAA (IEC 61850-7-2, 7.4)	☐ Passed ☐ Failed ☐ Inconclusive
IEC 61850-7-2 clause 7.4 IEC 61850-8-1 clause 10.2 PIXIT		
Expected result 1. SUT accepts Associate.response+ from server 2. SUT returns to “state” where it is able to start a new TPAA with the same server		
Test description 1. Set-up a TPAA with one server 2. Force SUT to release or abort TPAA 3. Repeat step 1 and 2, 10 times		
Comment		

Figure 1. Test procedure "Client Association 1" [1]

2. Conformance Testing Methodology and Framework

Standardization organizations like ISO, IEC, ITU-T or ETSI are not only creating standards but are also concerned with verifying the conformance of a given implementation to them. This led to the development of the Conformance Testing Methodology and Framework (CMTF) which is published as the ISO/IEC 9646 / ITU-T X.290 to X.296 series of standards, respectively. Part 3 of this series introduces the Tree and Tabular Combined Notation (TTCN), the ancestor of today’s Testing and Test Control Notation (TTCN-3), which is at the center of our attention in this paper.

ISO/IEC 9646 precisely defines a common vocabulary and methodology for how to derive conformance tests for a given specification in a multi-step process. In strong compliance with this general approach, the ETSI Technical Committee “Methods for Testing and Specification” (TC MTS) developed the technical specification ETSI TS 102 351 “IPv6 Testing: Methodology and Framework” [2]. It describes the usage of ISO/IEC 9646 to develop abstract TTCN-3 test cases for conformance and interoperability testing of IPv6 standards but it also states that “*it could be equally used in other areas of protocol test specification*”. [2, p. 6]

The procedure begins by collecting all requirements from standards, specifications and de-facto practices into a *Requirements Catalogue*. Then, at least one *Test Purpose* (TP) is written for each requirement. Such a TP describes what has to be tested as well as the initial condition before the testing and the criteria on which the verdict is based upon. However, the TP should not describe how the test has to be carried out. In order to ensure that TPs are written in a similar and consistent way, ETSI defined a structured format called *TPLan*. It consists of a small set of well-defined keywords and a simple syntax specifically targeted to express TPs for conformance and interoperability tests. The *Test Suite Structure* groups the TPs according to some logical criteria, e.g., “normal behavior", “exceptional behavior”, etc.

In the last step, abstract test cases are written in the TTCN-3 language which is described below in section 2.a. The TTCN-3 code is split into two logical groups, the *Library* and the *Abstract Test Suite Repository*. The Library contains reusable generic elements (e.g., data definitions, matching templates, recurring component behavior). All functions that are specific to a Test Suite are stored in a Test Suite Repository. These functions are the Test Cases, accompanying functions, Preamble and Postamble. Ideally they do hardly more than invoke functions or use data and templates for the Library. The final result is the Abstract Test Suite Repository with Test Cases for the given standard or specification.

It has to be noted that the artefacts produced in the individual steps (requirements catalogue, TPs, etc.) are very valuable results on their own and significantly contribute to a quality-oriented, traceable test development process.

2.a TTCN-3

TTCN-3 has been successfully employed in several domains (e.g. telecommunications, automotive, avionics, health) and by several standardization bodies and consortia (ETSI, 3GPP, OMA, Wimax Forum, AUTOSAR) over the last 15 years. To the best of our knowledge, there has interestingly been hardly any usage in the domain of power systems (one exception being [3] where a TTCN-3 test system was used to conduct tests of an IEC 61850-based protection device). To further investigate the concrete applicability of TTCN-3 for conformance testing of IEC 61850 we implemented existing

UCAIug client tests [1] in TTCN-3 and executed them against a commercial implementation. This activity is reported in section 3 after a brief introduction to TTCN-3.

TTCN-3 is a formal language which is specifically tailored to testing. As such it provides constructs that are not found in general purpose programming languages. An example are language elements for data matching that are commonly required for comparing incoming data against test expectations. The language provides an extensive type system and it can also make use of ASN.1 and XML schema types (and JSON in the near future) as well as IDL-based interface definitions. TTCN-3 provides constructs for message-based and procedure-based communication and enables the concurrent execution of parallel test components in a simple way. There are TTCN-3 extensions related to, e.g., performance and real-time testing, as well as interfaces with continuous signals.

The TTCN-3 test cases are independent of the implementation language of the test system and the implementation language of the SUT. TTCN-3 is not "just" a language but instead provides a complete test system architecture with standardized interfaces among its components as depicted in Figure 2.

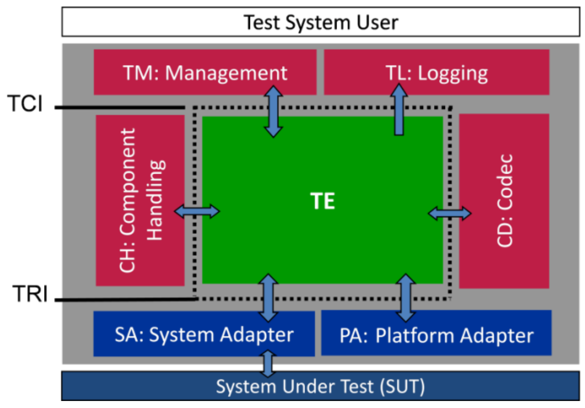

Figure 2. The TTCN-3 test system architecture [4]

The TTCN-3 test cases are compiled into executable code, which, in combination with a TTCN-3 runtime system, form the core Test Executable (TE). It is worth pointing out that the TE alone cannot be used to test a concrete SUT. The TTCN-3 test architecture provides for a clear separation between the abstract tests cases and the specifics of a particular SUT through the Test Runtime Interface (TRI). On the TTCN-3 test case level there are generic procedure- or message-based communication primitives exchanging platform-independent TTCN-3 data types. This approach enables the specification of reusable test cases that are not tied to the specifics of a given SUT.

To enable the testing of a concrete SUT, codecs (CD), a System Adapter (SA) and a Platform Adapter (PA) have to be provided. The codecs implement the mapping between TTCN-3 data types and the native data format of the SUT. The PA provides timers and other platform specific components. The SA establishes the communication with the SUT and thereby bridges the gap between the abstract test system interface (TSI) and the SUT. It may seem obvious but it is worth pointing out that this requires that i) an API is provided by the SUT and ii) the TSI can be implemented by the means of the native API of the SUT.

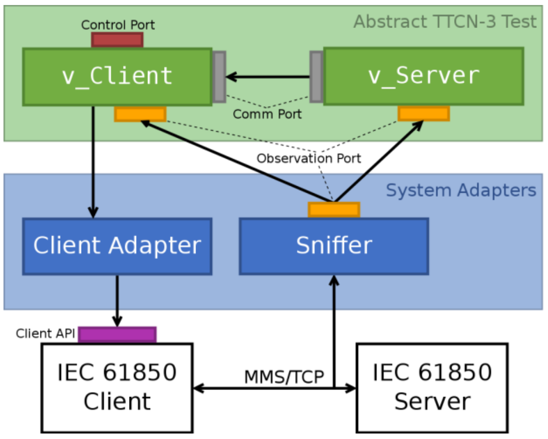

Figure 3. Test architecture for the IEC 61850 client conformance tests

Overall, the core TE combined with the SUT-specific artefacts CD/PA/SA provide for test cases that can be executed in a fully automated fashion. The execution is controlled via the test control interface (TCI) and components for test management and logging.

3. IEC 61850 client conformance tests in TTCN-3

For a hands-on investigation, we implemented TTCN-3 test cases for a subset of the IEC 61850 client tests as defined by the UCAIug [1]. All tests were executed against a level-A certified client provided by COPA-DATA. This client connects to a equally certified IEC 615850 server. The client provides an API to trigger certain actions and query its status (e.g., trigger an application association and then query the association status). We developed a system adapter that makes this API accessible via TTCN-3. On the server side, a network sniffer captures all network traffic and forwards it to the test runtime. The setup is depicted in Figure 3. Figure 4 shows the relevant parts of an abstract test case for a client association test in TTCN-3 (see Figure 1).

Two *test components*, `v_Server` and `v_Client` act as TTCN-3 proxies for the server and client, respectively. They do not implement any real functionality but simply exchange messages within the runtime environment or with the actual implementations via a system adapter using so called *ports*. In our setup we have three ports, `pt_ObservationPort`, `pt_ControlPort` and `pt_CommPort`.

Both components receive messages from the sniffer via `pt_ObservationPort`. The sniffer's System Adapter uses the source IP address to forward the messages to the correct component. Messages with the source IP of the client are sent to the client component and messages with the server's source IP are sent to the server component. The codec maps from the on-wire data formats to the TTCN-3 data types.

The client component can access the client's API via `pt_ControlPort`. The client's system adapter takes the `requestPdu` (e.g., a MMS *initiate-RequestPDU* in case of the client association test case using the MMS SCSM) and calls the correct client API function.

```
testcase Ass1_Associate() runs on MTC system TSI {
    clientAddress := {ipAddress := "192.168.1.1"}
    serverAddress := {ipAddress := "192.168.1.2"}
    f_configClientServer(clientAddress, serverAddress);

    v_Server.start(f_Replay());
    v_Client.start(f_Associate(requestPDU, responsePDU));
    f_waitAndGuard();

    f_deconfigClientServer();
}

function f_Associate(in 61850Pdu requestPdu,
  template 61850Pdu responsePlusPdu) runs on ClientComponent  {
    pt_ObservationPort.send(Command:startTrace);
    pt_ControlPort.send(requestPdu);

    // wait for the association request packet
    alt {
        [] pt_ObservationPort.receive(requestPdu) {
            setverdict(pass);
        }
        [] pt_ObservationPort.receive(61850Pdu:?) {
            log("Ignoring Message");
            repeat;
        }
    }
    // wait for the association response packet
    alt {
        [] pt_CommPort.receive(responsePdu) {
            setverdict(pass);
        }
        [] pt_CommPort.receive(negativeResponsePdu) {
            setverdict(fail);
        }
        [] pt_CommPort.receive(61850Pdu:?) {
            log("Ignoring Message",);
            repeat;
        }
    }

    pt_ObservationPort.send(Command:stopTrace);
}
```

Figure 4. Section of a TTCN-3 test for a client association

The last port, `pt_CommPort`, connects the server component and the client component within the TTCN-3 environment. The server component uses that port to send the packets it received from the sniffer to the client.

Neither the server component nor the communication port are strictly necessary as the observation port could be used for traffic in both directions. However, the chosen setup has the advantage that it is easier to understand the code during the implementation phase and the logging output of the test case execution.

The function `f_configClientServer` sets up the client and server and initializes the ports, `f_deconfigClientServer` releases the ports and resets the client and server.

After configuration, the server component runs `f_Replay`, a function that simply receives messages from the observation port and forwards them to the client component via the communication port. The client component runs the function `f_Associate`. It takes two arguments: the `requestPdu` as the association request message that has to be sent by the client and the template `responsePdu` to match the expected response. Internally, it also uses `negativeResponsePdu`, a globally defined template for negative responses. The function first starts the sniffer by sending the command `startTrace` via the observation port and triggers the association by sending the request message via the control port. Subsequently, it waits until it receives a message that matches the request PDU on the observation port and a message that matches the response PDU template or the negative response PDU on the communication port. Other 61850 messages are simply ignored.

The test is successful (i.e. the verdict is *pass*) if the matching messages are received in the correct order; the verdict is *fail* if the second message is a negative response. Otherwise the test will remain in one of the `alt`-statements, which list all possible alternative execution paths. Finally the function `f_waitAndGuard` implements a timeout to prevent a livelock or deadlock situation. It waits until either the client component is done with the execution of `f_Associate` or a predefined timeout is reached. If that happens, it terminates the test case and the verdict is set to *inconclusive*. The graphical log output for the client association test case is shown in Figure 5.

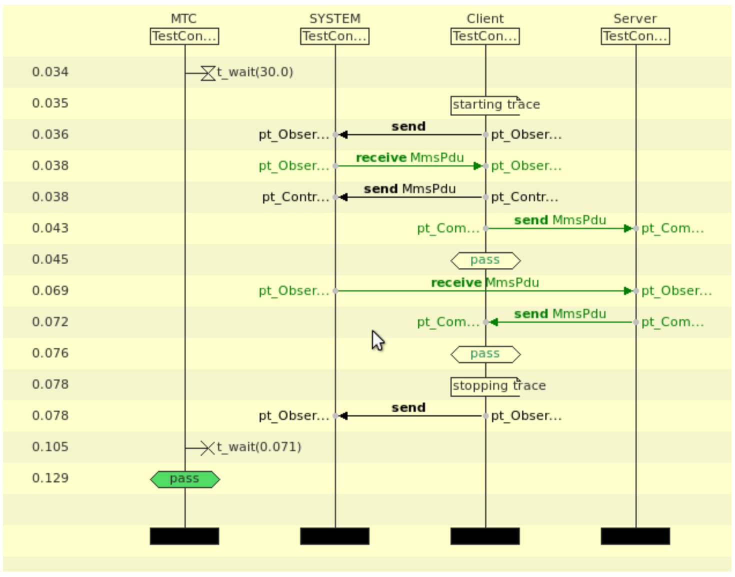

Figure 5. Graphical log output for the client association test case

It is very easy to re-use and extend this general test setup for other client tests that do not require any direct interaction between the test system and the IEC 61850 server, i.e., test cases for 'normal' operation. We have in fact successfully implemented several test cases including unbuffered and buffered reporting and direct as well as select-before-operate control models. The creation of tests was generally significantly simplified through the ability to directly use the ASN.1 definitions of MMS as the TTCN-3 types. The implementation of proper initialization and cleanup functionality in the system adapters enabled a fully automated test execution. The SUT-specific aspects are exclusively located in the System Adapter. Testing a different SUT would be straight forward as it would merely involve a new implementation of the client adapter (see Figure 3).

To test a client's behavior against a server that exhibits non-conformant behavior a special test server that can be configured for such a behavior would be needed. Such a server was not available to us and we could thus not test such cases. If it were, we could have integrated it into the TTCN-3 environment such that the server configuration can be set as part of a test case. Alternatively, one could create a mock server that mimics the relevant server behavior completely in TTCN-3. However, implementing such a server was clearly beyond the scope of our study.

4. UML-based test development

Creating abstract TTCN-3 test cases is clearly one way of providing far more formalized test specifications than prose test descriptions. An alternative approach would be the use of UML. In the recent years, efforts have been made to model IEC 61850 in UML [5,6,7]. The main objectives have been to improve the consistency and overall quality of the data model; to auto-generate parts of the standard from the model; to enable web-based access to the model; and to provide a machine-readable XML file of the model that can be used for semi-automatically checking the compliance of SCL configuration files [7,8].

To the best of our knowledge, formalizing the process of test development has not been a driving force of the UML development. However, a complete UML model of the IEC 61850 domain and its communication services would suggest itself for at least considering test development on top of it. Along the same argument, [8] mentions the possibility of using UML - and in particular the UML Testing Profile (UTP) [9] - for test development.

On the plus side, this approach would guarantee consistency of the test development with the system model as the classes and communication services of the IEC 61850 model can be referenced from the UTP tests. Also, the structural elements of the base UML domain model can be used in a straight forward way to derive the test architecture, i.e. the test components, ports, and data types, out of it.

However, UTP-based test development also has several drawbacks compared to the TTCN-3 approach:

- Although the UTP-based approach generally may seem to be on a higher level of abstraction compared to TTCN-3 tests this is particularly not the case as the sequence diagrams used to define the test behavior in UTP demand the same level of detail.
- Compared to TTCN-3, UTP provides limited capabilities for test definitions. Therefore, it is possible to automatically map UTP tests to TTCN-3 [10,11] but not the other way around. TTCN-3 enables the specification of test dynamics that are not possible in UTP, e.g. the dynamic creation of test components.
- Graphical modeling [12], as done in UTP, is also available through the TTCN-3 graphical format (GFT) [13], which is based on Message Sequence Charts 2000 as defined in ITU-T Recommendation Z.120 [14]. The benefit of the GFT is that it enables full round-trip-engineering with the textual TTCN-3 format in contrast to the one-way mapping from UTP to TTCN-3.
- UTP provides no means to specify complex test data as required in modern communicating systems, whilst one of the great advantages of TTCN-3 are the extended capabilities to define test data including powerful matching mechanisms.
- UTP test cases are not executable and they have to be translated to some other language to obtain executable artifacts. UTP provides a mapping to JUnit and TTCN-3, respectively.
- Unlike UTP, TTCN-3 is still actively maintained and further developed under the auspices of ETSI. Version 4.7.1 of the core language is expected to be published in June 2015.

From our point of view and practical experience these drawbacks are severe. We are thus is favor of the TTCN-3 approach to formalizing test specifications.

5. Conclusion

From the recent efforts of creating a UML model for IEC 61850 we deduce that there is generally an ambition to progress towards more formalized standards and its corresponding conformance tests. To this end we have studied existing approaches taken by standardization consortia in the telecommunication and other domains. The Conformance Testing Methodology and Framework (CMTF; ISO/IEC 9646 / ITU-T X.290 to X.296) is an industry proven methodology to create formalized conformance test specifications. In the process of defining test cases several valuable intermediate results are created, like the requirements catalogue and test purposes written in the special purpose TPLan language. Finally, the abstract test cases are formalized in TTCN-3.

From a technical point of view, we argue that there are several significant benefits in providing abstract test cases in TTCN-3 compared to prose descriptions. The standardized TTCN-3 language with its well-defined semantic helps eliminate the ambiguities of English (or any other language for that matter) and, consequently, a lot of room for interpretation. It allows for precise specifications of the expected static and dynamic behavior and provides various test-specific language construct like e.g. a full-featured so called template system that enables brief and concise data matching statements. As TTCN-3 can make use of ASN.1 and XML Schema types, as well as IDL-based interface definitions, the type specification of standards can often be used directly in (or at least imported into) the TTCN-3 environment.

If abstract TTCN-3 tests were provided to the community (as is, e.g., the case for IPv6 conformance tests), they would serve as a common, reusable starting point for the many testing activities carried out by the various industrial players. Still, there is no vendor lock-in and testers would not be restricted in their choice of testing tools and/or implementation languages, respectively.

As an alternative, formalizing conformance tests via UML/UTP is feasible and such tests can be automatically mapped to TTCN-3 in order to obtain executable test cases. However, despite being an intermediate step, the UTP approach does not provide a higher level of abstraction than TTCN-3 and has comparatively limited test modelling capabilities.

From a non-technical point of view, we argue that an open standard of a formal test specification and execution language is preferable to prose test descriptions and their derived proprietary and closed test implementations. Especially for a technology that is used to control parts of a critical infrastructure it seems desirable to create open test specifications that decrease the room for interpretation and at the same time increase reusability and ultimately confidence and trust throughout the industry.

As a next step we are currently working on tests with stringent performance requirements and physically distributed test components. The use case is a load-shedding procedure based on synchrophasor measurements that are transmitted via a recently implemented IEC 61850-90-5 stack.

Acknowledgement

We would like to thank Copa-Data for their valuable feedback and support in conducting this study.

This work is partially funded by FFG (Österreichische Forschungsförderungsgesellschaft) through the ERA-SME project OTITOS.

References

[1] M. Flohil and R. Schimmel, “Conformance Test Procedures for Client System with IEC 61850-8-1 interface. Revision 1.1,” KEMA Consulting, 2009.

[2] ETSI TS 102 351 v2.1.1, “Internet Protocol Testing (IPT); IPv6 Testing: Methodology and Framework”, August, 2005

[3] S. Schwabe et al: Synchronized Distributed Testing using TTCN-3 and GPS, TTCN-3 User Conference 2007, 29 May - 1 June 2007, Stockholm, Sweden

[4] Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 5: TTCN-3 Runtime Interface (TRI), ETSI Std. ETSI ES 201 873-1 V4.6.1, 2014.

[5] T. Kostic, O. Preiss, C. Frei, "*Understanding and Using the IEC 61850: A Case for Meta-Modelling*", Elsevier Journal of Computer Standards & Interfaces, vol. 27/6, pp. 679-695, 2005.

[6] O. Preiss, T. Kostic, C. Frei, "*Data Communications Standards: A Case for the UML*", Lecture Notes in Computer Science, Volume 3297 / 2005, <<UML>> 2004 Satellite Activities, N.J. Nunes et al. (Eds.): pp. 175 - 186, February 2005.

[7] Apostolov, A., "UML and Its Use," Presentation to the IEEE PES PSRC, 13 January 2014, New Orleans, USA

[8] Laurent Guise et al. "*UML - An on-Going Great Step Forward in Improving IEC Smart Grid Standardisation Processes*", PAC World 2012, Budapest, Hungary

[9] “UML Testing Profile (UTP)”, [Online]. Available from: http://utp.omg.org, 2015.04.29.

[10] I. Schieferdecker, Z.R. Dai, J. Grabowski, A. Rennoch. *"The UML 2.0 Testing Profile and its Relation to TTCN-3"* Testing of Communicating Systems. pp. 79-94. Proceedings of the 15th IFIP International Conference on Testing of Communicating Systems (TestCom2003), Sophia Antipolis, France, May 2003. Lecture Notes in Computer Science 2644. Springer-Verlag GmbH 2003.

[11] J. Zander, Z.R. Dai, I. Schieferdecker, G. Din. *"From U2TP Models to Executable Tests with TTCN-3 - An Approach to Model Driven Testing".* Proceedings of TestCom2005, Montreal May 2005.

[12] I. Schieferdecker and J. Grabowski, “*The graphical format of TTCN-3 in the context of MSC and UML*” in Telecommunications and beyond: The Broader Applicability of SDL and MSC, ser. Lecture Notes in Computer Science, E. Sherratt, Ed. Springer Berlin Heidelberg, 2003, vol. 2599, pp. 233–252.

[13] Methods for Testing and Specification (MTS) “*The Testing and Test Control Notation version 3; Part 3: TTCN-3 Graphical presentation Format (GFT)*”, ETSI Std. ETSI ES 201 873-3 V3.2.1, 2007.

[14] ITU-T Recommendation Z.120 (2000): “*Message Sequence Charts (MSC)*”, 2011