TTCN-3, Qtronic and SIP

The Model-Based Testing of a Protocol Stack — a TTCN-3 Integrated Approach

Technical Whitepaper

EXECUTIVE SUMMARY

TTCN-3 (Test and Test Control Notation version 3) is a programming language for developing tests in the telecommunications domain. **SIP** (Session Initiation Protocol) is a key protocol standard in the modern convergent telecommunications sphere. **UML** (Unified Modeling Language) is one of the most widely used modeling notations for describing requirements and design beyond natural language. **Java** is one of the most popular programming languages as of today. **Conformiq Qtronic** is a tool for generating tests automatically from system design models, developed by Conformiq Software Ltd.

We tested a publicly available SIP stack using both **model-based testing** and TTCN-3. We started by examining the publicly available standard for SIP. We then created a corresponding design model in UML and Java based on the SIP standard alone, and implemented a system adapter for connecting a TTCN-3 runtime environment to both the control and peer-to-peer interfaces of the SIP stack. Using the Conformiq Qtronic tool, we generated tests from the design model and produced a TTCN-3 test suite, which we then executed against the SIP stack with the aid of a TTCN-3 tool.

In summary, we were able to employ model-based testing in a TTCN-3 environment. **We were able to fully automatically derive TTCN-3 test cases from a functional design model and to execute them against a real system.** This shows that model-based testing can be employed successfully in a TTCN-3 driven quality assurance process using the tools available from Conformiq Software Ltd.

Keywords: Model-Based Testing, TTCN-3, SIP, UML

Conformiq Software Ltd. | Tekniikantie 12 | FI-02150 Espoo, FINLAND | Tel. +358 10 286 6300 | Fax +358 10 286 6309
www.conformiq.com

1 INTRODUCTION

In this whitepaper we explain how we tested a publicly available protocol stack using **model-based testing** and TTCN-3. **TTCN-3** (**Test and Test Control Notation version 3**) is a notation for describing executable test cases, standardized by **ETSI** (**European Telecommunication Standards Institute**). TTCN-3 is nowadays widely applied in the telecommunications domain.

The protocol stack in question is the **Sofia stack for SIP**, developed originally at Nokia Research Center (NRC) but nowadays publicly available. **SIP** stands for **Session Initiation Protocol**. It is a standardized protocol (standards body **IETF**, the **Internet Engineering Task Force**) for establishing, managing and terminating media sessions and handling the necessary infrastructure such as registering the physical terminals that are available for users. It is widely used for example in the **VoIP** (**Voice over Internet Protocol**) domain.

Model-based testing means test automation where executable tests are automatically generated from system **design models**. It extends the notion of test automation from mere test **execution** to also test **design**. It has therefore potential for reducing quality assurance risks and costs, because it removes human effort from the areas of **test design** and **test selection**. These areas are not usually addressed by other test automation concepts.

A model-based testing solution must integrate as seamlessly as possible with the existing processes. In this whitepaper our aim is to demonstrate how model-based testing works in a testing process that is based on the use of TTCN-3.

2 THE TESTING SETUP

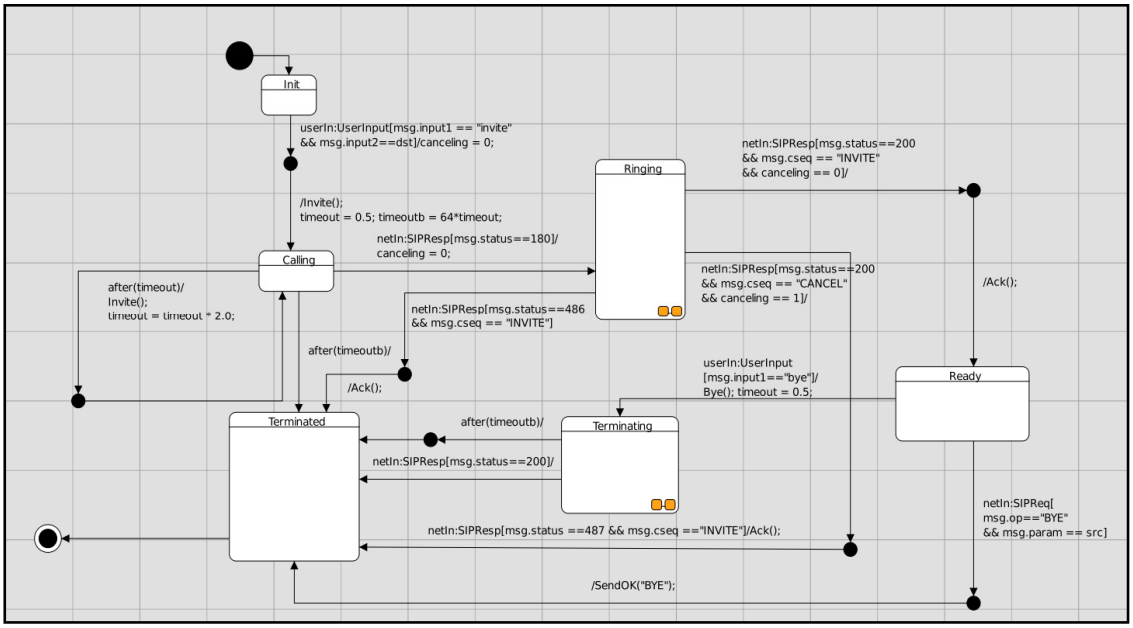
The Sofia stack is a C implementation of SIP. It has naturally two testing interfaces: the control interface, which is a C API, and the peer-to-peer interface, which is a UDP context carrying SIP messages.

2.1 Design Model

We constructed the design model using **UML statecharts** and **Java** as the design language. The UML statecharts were drawing using **Qtronic Modeler**, a simple **XMI** (**XML Model Interchange**) enabled model editor. Fully textual parts of the model were written into text files using a normal text editor (**GNU Emacs**).

Conformiq Qtronic supports this language (UML + Java) for expressing executable (platform independent) designs. However, it supports also others.

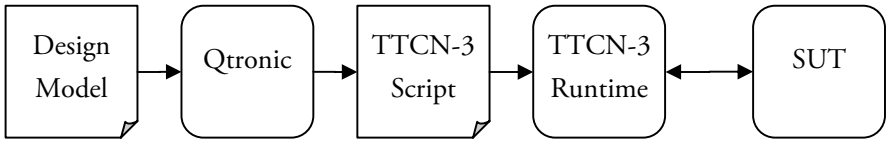
We created our design model by using the SIP standard (RFC 3261) as the reference. In particular, the implementation (Sofia SIP stack) was not consulted when creating the design model. This is ensured that testing was independent of the implementation.



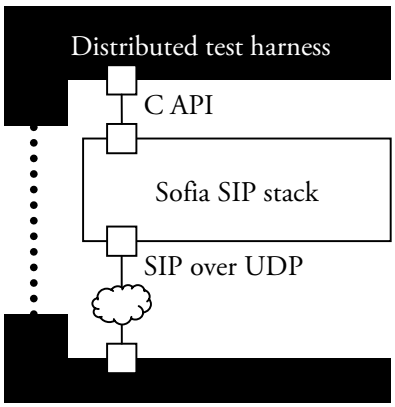
The main part of the UML statechart for SIP UAC in our setup.

2.2 Test Generation Workflow

We generated the tests from the SIP design model that we created by ourselves. We generated the tests using **Conformiq Qtronic**, our tool for model-driven test generation. The tool reads in a design model and generates test scripts using configurable back-ends. There are various options in the tool for controlling the size and characteristics of the scripts generated. Conformiq Qtronic is also capable of **online testing**, i.e. executing tests directly against a **SUT** (**System Under Test**) without going via test script generation.



2.3 Test Harness



In our setup we created a **test harness** in a distributed fashion so that the SIP protocol UDP context is actually established between two physically different network nodes. This made it easier for us to monitor the SIP traffic using tools like **tcpdump**.

The test harness was integrated with an existing TTCN-3 testing solution, the commercially available **TTWorkbench** from **TestingTech**. However, no aspect of the solution is specific to this TTCN-3 environment. The TTCN-3 code that was generated, for example, is completely standards-compliant and portable across different TTCN-3 tools.

We focused on the behavior of the Sofia stack as an **UAC** (**User Agent Client**) when it is trying to establish, manage and terminate sessions.

2.4 On-the-Fly Testing During Harness Implementation

The test harness was actually constructed so that the same adapter could be used with both TTWorkbench as well as the adapter API for the **on-the-fly testing feature** of Conformiq Qtronic. In addition to generating test scripts, Conformiq Qtronic can also run tests directly ("on the fly") against the system under test. Whereas it takes some time and lots of manual maneuvers to generate TTCN-3 and run it through TTWorkbench, on-the-fly testing commences from Conformiq Qtronic with a single mouse click. Because the adapter could work with both the tools, its development was made significantly faster by using on-the-fly testing during adapter development.

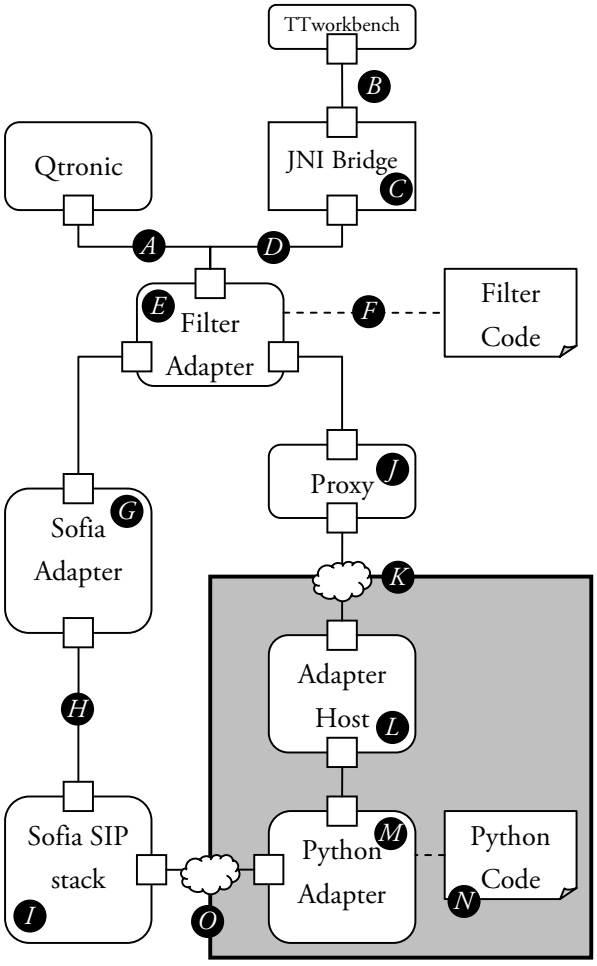
It is worthwhile to understand the fundamental difference between test script generation and on-the-fly testing, even though this subject is beyond the main topic of this whitepaper. Because test scripts are generated offline, it is impossible to guess the run-time behavior of the SUT while constructing the scripts. If the SUT has multiple correct behaviors test script generation becomes quite difficult.

Therefore the test script generation feature of Conformiq Qtronic currently focuses on systems that should work in the same way whenever subject to the same testing inputs. Now in the SIP protocol there are fields that can contain numbers randomly chosen by the UAC. Because these random numbers are not known during script generation time, they cannot appear in the test suite at all. Therefore the handling of these randomized sequence numbers was moved to the test adapter in our solution.

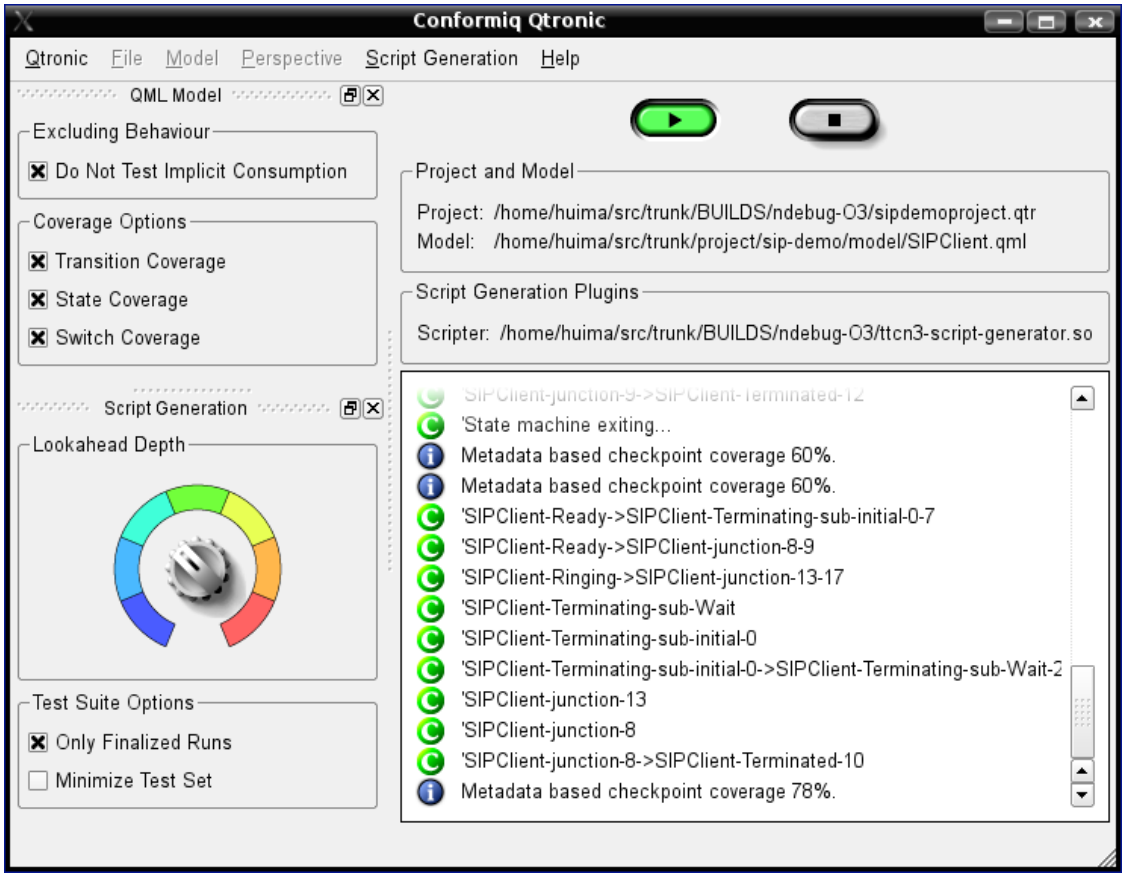
If on-the-fly testing would have been solely used, then the sequence numbers could have been handled in the model itself because on-the-fly testing does not have the fundamental limitation explained above. In on-the-fly testing the tester can examine the outputs from the SUT during

testing and adapt the next test inputs to them. This is possible because the whole test sequence has not been constructed beforehand—a contrast with offline script generation.

We show the whole testing architecture on the right for the benefit of the technically inclined. For on-the-fly testing, **Qtronic** connects to a system adaptation component using a **C++ plug-in API** (*A*). **TTWorkbench** instead is based on Java and uses the **class loading mechanism** to load adapters (*B*). In order to reuse Qtronic adapters in TTCN-3 environment we created a Java component (*C*) that, employing **JNI** (Java Native Interface), can talk (*D*) with a Qtronic C++ plug-in. The first Qtronic adapter is a generic **filter component** (*E*) that can dispatch messages to multiple sub-adapters and that can be programmed (*F*). The sub-adapters use the same Qtronic C++ plug-in API. The filter is used to eventually send the messages to the peer-to-peer and control interfaces via two different hosts. One of the sub-adapters is the adapter (*G*) to the **Sofia stack** (*I*). The connection (*H*) is via the **Sofia C API**; the adapter is linked with the Sofia library. The other sub-adapter is a proxy (*J*) that enables one to **distribute** adapters in internet. The proxy communicates over TCP/IP (*K*) with an **adapter host** executable (*L*) at another host (the gray box). The adapter host provides the same plug-in API as Qtronic, into which an adapter providing a **Python interpreter** is attached (*M*). The adapters loads Python code (*N*) that implements the peer-to-peer part of the adaptation, completed with a **UDP connection** to the SIP stack under test (*O*). The generated TTCN-3 is **independent** of the particular testing architecture, though. By sharing the same adapter with Qtronic and TTWorkbench, we could exploit on-the-fly testing during adapter programming.



Conformiq Software Ltd. | Tekniikantie 12 | FI-02150 Espoo, FINLAND | Tel. +358 10 286 6300 | Fax +358 10 286 6309
www.conformiq.com



3 GENERATING THE TEST SUITE

We generated the test suite using Conformiq Qtronic. Test generation took 16 seconds on a modern desktop PC. The resulting test suite contained 18 test cases containing altogether 160 events (messages between the test harness and the SUT). The generated test suite was guaranteed to cover 84% of the states, transitions and switches in the original design model.

3.1 Generated TTCN-3

The underlying test scripts generated by Conformiq Qtronic are sequences of timed messages. The mapping of these sequences to TTCN-3 is mostly straightforward. The main issue is the handling of time. For example, in the SIP protocol there is a timer that causes retransmissions of the INVITE message first after 500 ms, then after 1 second, then 2 seconds, the timeout doubling in length every iteration. Conformiq Qtronic correctly reflects this in the constructed test sequences. The problem is that in practice these resends cannot appear at the TTCN-3 executor *exactly* at these times, but they appear slightly later due to lags in the UDP transport and the various software components of the

testing setup. The generated TTCN-3 is therefore constructed so that it accepts certain delay in receiving the expected messages, and adjusts the timing of the remaining part of the test case accordingly if needed.

3.2 Controlling Generation

The set of test cases is chosen according to **model-based test coverage**. A component of the model (for example, a method, a transition or a conditional branch) can be said to be **covered** by a test case if executing the test case against the model would guarantee that the said component would be visited during the test. Conformiq Qtronic attempts to create a test suite that covers as many of the different model components of those types the user has selected via the tool's user interface.

This approach creates a reasonably compact test suite that exercises a large amount of the functional features in the model. Further control over the generated test suite can be then obtained by restructuring the model. It is even possible to create a design model that actually encodes specific **use cases** or **test purposes**.

4 RUNNING THE TESTS

We compiled and run the tests against the Sofia SIP stack through our distributed test harness. Executing all the tests took 1 minute and 10 seconds. 100% of the test cases concluded successfully, an expected result given the expected stability of the product and the focus of the testing which was on the basic, core functionality of SIP. Obviously the tests generated do not cover all aspects of SIP, but only those modeled. But this also shows that model-based testing can be used in an incremental fashion: even a partial and abstract design model creates concrete value as a testing asset.

5 THE VALUE OF THE APPROACH

The model-based testing approach creates a direct link from design to quality assurance. It adds new value to design modeling: in addition of their intrinsic value as documentation, design models become also testing assets. Model-based testing also removes risks and costs because separate testing assets (test scripts) that require maintenance decrease in number.

The integration with TTCN-3 brings process-level benefits in contexts where TTCN-3 forms an integral part of the quality assurance process. Automatically generated test scripts can be stored in configuration management / version control systems, and they can be executed even when the model-based testing tool is not available.

Conformiq Software Ltd. | Tekniikantie 12 | FI-02150 Espoo, FINLAND | Tel. +358 10 286 6300 | Fax +358 10 286 6309
www.conformiq.com

6 MORE INFORMATION

To receive an electronic copy of this whitepaper, please contact us at *sales@conformiq.com*.

Here are some links to further information on the subjects mentioned in this whitepaper:

Conformiq Qtronic	**www.conformiq.com**
Conformiq Software Ltd.	**www.conformiq.com**
SIP	RFC 3261 and others (www.ietf.org)
Sofia SIP stack	opensource.nokia.com/projects/sofia-sip/index.html
TTCN-3	www.ttcn-3.org
UML	www.uml.org
Java	java.sun.com
XMI	www.omg.org/technology/xml