Electronic Notes in Theoretical Computer Science 80 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume80.html 10 pages

Tool Supported Test Frameworks in TTCN-3

Ina Schieferdecker [1]

Fraunhofer Research Institute for Open Communication Systems (FOKUS)
Berlin, Germany

Theofanis Vassiliou-Gioles [2]

Testing Technologies IST GmbH
Berlin, Germany

Abstract

This paper presents the concept of test frameworks and the specification and implementation of test frameworks in TTCN-3 being the Testing and Test Control Notation for black-box and grey-box tests of reactive systems. Test frameworks enable an easy reuse, extension and customization of predefined test patterns for test data, configuration, behavior and control. Test patterns can be combined and customized to yield additional tests for further system features, changed system characteristics as well as for additional test requirements. The implementation of test frameworks and the execution of resulting tests are supported by a TTCN-3 tool chain providing means for graphical test specification, compilation, management and execution. A GPRS example demonstrates the practical application of test frameworks using a set of TTCN-3-based tools.

1 TTCN-3 Overview

The Testing and Test Control Notation TTCN-3 [1,2,3,4,5] is the new test specification and test implementation language that supports all kinds of black-box testing of distributed systems such as functional, interoperability, regression, scalability, load, conformance, etc. tests. TTCN-3 was developed in the years 1999 to 2002 at the European Telecommunications Standards Institute (ETSI), as a redesign of the Tree and Tabular Combined Notation (TTCN) standard. It is being published as the ETSI standard series ES 201 873 and also as the ITU-T Rec. Z.140 series and is under ongoing continuous maintenance at ETSI. TTCN-3 is built from a textual core language that

[1] Email: schieferdecker@fokus.fraunhofer.de
[2] Email: vassiliou@testingtech.de

provides interfaces to different data description languages and the possibility of different presentation formats.

TTCN-3 is a flexible and powerful language, applicable to test reactive systems via a variety of communication interfaces. Typical areas of application are protocol testing (including mobile and Internet protocols), service testing (including Supplementary services and Web services), module testing, testing of middleware components (including CORBA ORBs and CORBA based systems), and the testing of APIs.

Retaining and improving basic concepts of predecessors of TTCN-3 and adding new concepts increase the expressive power and applicability of TTCN-3. New concepts are, e.g., a test execution control to describe relations between test cases such as sequences, repetitions and dependencies on test outcomes, dynamic concurrent test configurations and test behavior in asynchronous and synchronous communication environments. Improved concepts are, e.g., the integration of ASN.1, the module and the grouping concepts to improve the test suite structure, and the test component concepts to describe concurrent and dynamic test setups.

TTCN-3 allows an easy and efficient description of complex distributed test behavior in terms of sequences, alternatives, loops and parallel stimuli and responses. Stimuli and responses are exchanged at the interfaces of the system under test, which are defined as a collection of ports being either message-based for asynchronous communication or signature-based for synchronous communication. The test system can use any number of test components to perform test procedures in parallel. Likewise to the interfaces of the system under test, the interfaces of the test components are described as ports.

TTCN-3 offers various presentation formats to serve the needs of different TTCN-3 application domains and users. The programming-like textual core notation suits best programmers and test developers. The graphical format of TTCN-3 enables the visualization of test behavior, eases the reading, documentation and discussion of test procedures and is also well suited to the representation of test execution and analyzing of test results. The tabular presentation format highlights the structural aspects of a TTCN-3 module and in particular of structures of test data.

2 Test Frameworks in TTCN-3

Test frameworks are based on the understanding that the time for closed-box solutions in testing is over. Current systems to be tested are typically very complex in terms of size, features, distribution, involved components and used technologies. Even more, systems are also very dynamic in terms of configuration, control, usage contexts, component and system updates, optional features, addition of features, etc. Hence, a predefined, fixed set of tests is not able to cover the changing needs in testing. Rather, existing tests should be continuously updated along the changes in the system. Today's test solutions

should offer the ability to adjust tests to new testing requirements reflecting the system changes.

The principle of adjusting tests to updated requirements and changing system characteristics can be realized with different elements of a test specification being:

- Test data types and test data values
- Test configuration
- Test functionality and test cases
- Control of test case execution

as well as with different elements of the test execution. The ingredients for a test pattern based test framework in TTCN-3 are depicted in Fig. 1.

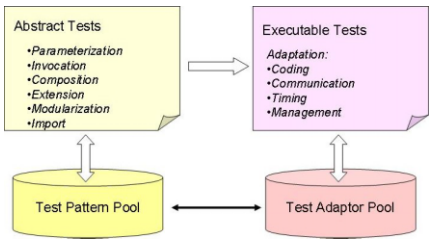

Fig. 1. Test Frameworks

Test pattern for test data, test configuration, test functionality and test control should be defined such that their extension, adaptation and composition are possible. TTCN-3 supports this with various language features such as the ability

- to parameterize different elements of a test specification such as templates, functions, test cases and modules,
- to combine test types to build complex structured test types and to combine test templates for structured types with an hierarchy of template (invocation)s,
- to structure test behavior into altsteps, functions and test cases and their parameterized invocations,
- to structure test control into functions and their parameterized invocations,
- to configure test setups into sets of test components, which can be dynamically adjusted to specific test execution needs, or
- to extend test specification elements with user-defined attributes.

Every predefined test specification element can be used as a test pattern.

If a test pattern is parametrizable[3] and extensible, it is generic and can be customized to specific needs. A basic idea of test frameworks is the provision of these test patterns via test pattern pools for their reuse in the definition of new tests. TTCN-3 supports this idea by providing fine granular import mechanisms for the import of selected test specification elements, i.e. of test patterns, from existing TTCN-3 modules.

A test framework does not only provide test patterns on specification level but also on execution level, as the base tests of the test framework (typically but not restricted to functional tests) are available ready for execution. A pool of test adaptors adhering to the test type and test configuration patterns in the test pattern pool and enabling the execution of tests defined in the test framework are to be provided as well. For TTCN-3 specifically, these adaptors follow the interfaces and operations [3,4] defined for the execution of TTCN-3 tests.

3 Tool Support

While the basic concepts of test frameworks, modularity and reusability are well-accepted in the software development process test frameworks as defined above are hard to find. One of the reasons was the lack of test specification features and of tool support for example in the area of telecommunication testing. With the development and availability of TTCN-3 as a multi-purpose test specification language, tool support for test frameworks is possible. This section outlines requirements on and properties of existing TTCN-3 tools that are addressing the concepts of test frameworks.

Tool support for test frameworks in TTCN-3 can be classified into three categories, tool support for test specification, for test execution and for migration. While the former two address the specification and implementation of new tests the latter addresses the exploitation and use of existing test resources for new test requirements and/or in new test environments.

With the graphical format of TTCN-3 (GFT) complete test specifications can be defined. For test frameworks specifically, GFT can be used to define additional test behavior on the basis of existing type and data definitions[4]. A GFT based test specification tool like Testing Technologies' TTspec should offer an easy access to existing type and data definitions not only in TTCN-3, but also for other supported languages in TTCN-3 like ASN.1 or IDL. As a native GFT tool, TTspec supports the usage of all language elements, syntactically and semantically, as specified in TTCN-3 [2]. In addition, it offers the selection and usage of imported types, data and behavior. Although

[3] TTCN-3 offers value parameterization only. However, with the support of the anytype, effects comparable to type parameterization can be realized.

[4] GFT represents test behavior graphically and assumes test types and data being defined in TTCN-3 core. It can also be combined with the tabular format of TTCN-3 (TFT): TFT for a graphical visualization of data and GFT for a graphical visualization of behavior.

the visualization and modification of such imported elements is a concern for the usage of a test framework, this type of functionality is of general relevance and less test framework specific. Even further, the modification of imported elements, for example of test types, has to be handled carefully as the test patterns being provided by the test framework are also used as a basis for test execution within the runtime environment of the test framework.

Resulting test specifications can be compiled with any TTCN-3 compliant compiler like Testing Technologies' TTthree in order to produce executable tests. No special requirements are imposed on a TTCN-3 compiler. However, as the executable test suite needs a TTCN-3 runtime-environment as well as adaptations to test devices and to underlying protocols, test framework specific requirements can be identified for TTCN-3 runtime environments.

Within the multi-part standard TTCN-3, two parts address the interworking between the generated code from TTCN-3 specifications and the TTCN-3 runtime environment [3,4]. Part 5 on the TTCN-3 Runtime Interface (TRI) addresses the implementation of adaptations specific to the system under test. Part 6 on the TTCN-3 Control Interfaces (TCI) addresses the implementation of technology specific adaptation like encoding/decoding and of test device specific adaptation like the test management. The relation between the abstract TTCN-3 test framework specification and the TRI/TCI adaptor implementations is that no modifications or extensions are needed on TRI/TCI adaptation level as long as specific elements of the test framework are used unmodified. Namely, type definitions used within communication and configurations like the test system interface and ports.

Obviously within a test framework not only the abstract TTCN-3 specification parts shall be extensible but also, if necessary, the adaptations within the target language. For example if data types have to be modified in order to respect proprietary protocol extensions, modifications of the encoding/decoding implementation might be necessary [5]. Another example is the evolution from functional test scenarios to scalability test scenarios. Here, new test configurations are introduced.

For this, a test framework must support convenient modification and extension of existing test adaptors. TTthree's runtime environment and interfaces adhere to this requirement [5,6]. It supports the Java language mapping of all interfaces as defined in [3,4]. This, together with a clearly defined, plug-in like extension mechanism lifts test frameworks to practical relevance. Consequently, users can conveniently extend provided test frameworks to fulfill their needs, both on abstract test specification and on test execution level for functional tests as well as for other kinds of tests.

Another aspect for the applicability of test frameworks is the availability of migration tools, either from previous TTCN versions or from other proprietary

[5] Whether adaptation of the encoding implementation is necessary, depends basically on the type of encoding rule and the available implementation.

test specifications to TTCN-3. While migration tools are not essential for basic test frameworks as such, their availability widens the applicability and acceptance of test frameworks as existing test resources can be added to the test framework, thus reusing previous investments and know-how.

For example, using a TTCN-2 to TTCN-3 translator like TTtwo2three enables the usage of elements of the plethora of TTCN-2 test solutions that are available. In particular, TTCN-2 type and data definitions can be imported to an existing TTCN-3 test framework and evolve a test framework from e.g. a pure functional test framework to an interworking test framework for tests between legacy and new technologies.

4 A GPRS Example

To illustrate the usability and efficiency of tool supported test frameworks in TTCN-3, a recently completed case study is presented in this section and the results are discussed.

The goal of the case-study performed together with a test device vendor was two-fold. On the one hand side, the integration of an existing test device into TTthree's TTCN-3 runtime environment had to be examined. On the other hand, the migration of existing test suites defined in TTCN-2 to TTCN-3 was analyzed. Within this case study, Testing Technologies' tool chain consisting of the TTCN-2 to TTCN-3 translator TTtwo2three, the GFT-based test specification tool TTspec as well as the TTCN-3 to Java compiler TTthree has been used. The target test device was a Tektronix K1297-G20 (in short G20). Fig. 2 displays the overall workflow of the case study.

The first task (1) consisted of the translation of an existing TTCN-2 GPRS test suite, that was already available on the G20 into TTCN-3. TTtwo2three translates an existing test suite into an equivalent TTCN-3 test suite, while keeping the semantics and structures as used in TTCN-2. This translation leads to a TTCN-3 test suite that consists of several modules. For example, one module contains type and configurations elements while another module contains data for communication, i.e. TTCN-3 templates. Typically, the translation of existing TTCN-2 test suites to TTCN-3 requires some manual interaction because of e.g. proprietary extensions to TTCN-2. Although this was also the case with the used TTCN-2 test suite, the problems encountered were only of minor relevance and resources spent for this manual work were neglectable.

Although the translation of the TTCN-2 test suite produced a complete TTCN-3 test suite including types, data and behavior, within the case study additional test cases have been defined using TTspec, the graphical editor for TTCN-3 (2). This served as an example how predefined types and data can used in TTspec in order to produce additional test functionality. Using the context-sensitive support of TTspec in defining test behavior, test cases have been designed and specified just by composing existing test patterns for data

and functions to new test cases.

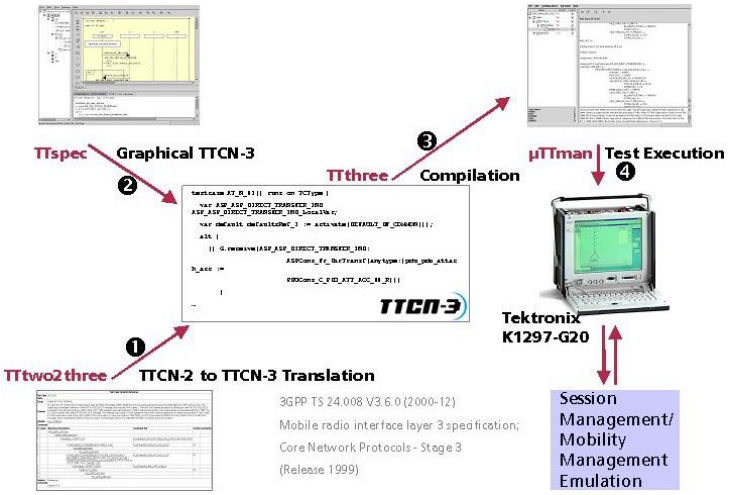

Fig. 2. GPRS Tests on the K1297-G20 Test Device

Fig. 3 displays a fragment of a test case defined using TTspec. TTspec provides context-sensitive support at the following places of the fragment:

- configuration (port types and names, test configurations, components) (1)
- references (2),
- message types and templates (3)
- timers (4)

Using the context-sensitive support, the visual definition of a test case could be achieved in a fraction of time. In addition, error-prone typing of identifiers could be avoided. The result of this task was a TTCN-3 module ready to compile by a TTCN-3 compiler.

The third task of the case study (Fig. 2/3) covered the implementation of the test suite for the Tektronix G20 using the TTCN-3 to Java compiler TTthree. For this task, the generic TRI and TCI base implementations of TTthree's runtime environment have been used. In particular, the TRI interface and its implementation have been used for realizing the adaptation to the G20, while the TCI value interface has been used to implement the encoding and decoding functionality. As a matter of fact, only the decoding functionality had to be implemented as the encoding of the respective protocol could be achieved using generic encoders provided by the runtime-environment. The complete implementation of the test suite took around five man-days [6].

[6] One target of the case-study was to realize a demonstrator that demonstrates the capabil-

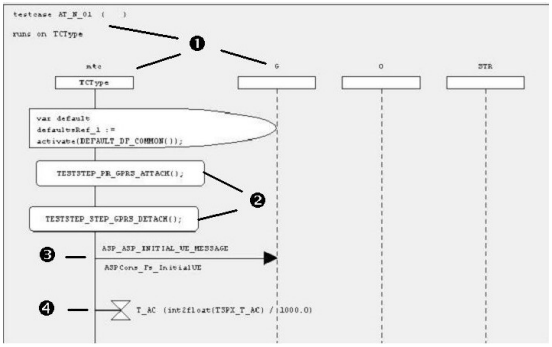

Fig. 3. GFT Fragment

At this point, a test framework had been developed according to the previous definitions. A set of test patterns in TTCN-3 (types and data resulting from task 1) and adaptors to the G20 test device and the encoding/decoding (result of task 3) were available. Then, the demand for the inclusion of additional test functionality arose after implementation of the adaptors. For the extension of this test framework, the development team used TTspec to define the additional test functionality as well as a text-editor to define additional test data, i.e. TTCN-3 templates in core notation. The nature of the required extensions did not impose modifications on the test setup or the type definitions (Fig. 2/2). No further modifications at the implementation level was necessary, neither on the TRI adaptation nor on the TCI based decoder.

The implemented test suite was executed using TTthree's build-in test management muTTman. muTTman is a generic test management tool that enables the execution of TCI-TM (test management) compliant test suites. The developed and enhanced test suite of the case-study could be executed without any further modifications.

For the case study, the following conclusions can be drawn:

- With existing tools providing open interfaces for adaptation and customization, the goals of this case study reflecting basic aspects of test frameworks can be achieved in an affordable way.
- The efforts necessary to adapt compiled test suites to a certain test devices depend heavily on the availability of well-defined programming interfaces on the test device side comparable to the TTCN-3 standardized programming interfaces. Within this case-study an almost ideal environment setting with respect to the test device has been anticipated: G20 provides a flexible,

ities of a TTCN-3 test solution using the G20. However, with the resources being available in the case study, nearly a prototype for a respective test framework has been achieved.

platform-independent, CORBA-based API.

- The majority of resources (around 2/3) were spent on the implementation of the decoding functionality. Here is obviously the highest potential for saving resources while adapting test suites for dedicated technologies. However, as the coding can be implemented completely independent from test devices adaptation (at least within the analyzed tool chain), coding is a major feature of practical test frameworks: the test framework provider offers the coding functionality as an integral part of the test framework. The user can use the built-in codecs directly for test execution and has to extend them in rare cases only.

5 Summary

In this paper we described TTCN-3, introduced the idea of TTCN-3 based test frameworks, and described and discussed the relevance of a tool chain that supports the notion of test frameworks. Within a case study, selected concepts of a test framework have been verified.

The idea of a test framework is derived from the need that the classical conformance testing approach supported by closed-box test solutions has decreased in relevance. Regulation does not require certification to the extend that was necessary a decade ago. Further, modern systems are open to extensions and changes. Testing has to reflect this changed setting.

Users are constantly seeking efficient and affordable tools in order to solve their testing needs. But as systems have become more flexible and the overall communication scenarios are more heterogeneous than ever, they have to have flexible testing tools. Test frameworks are a proper answer to this changed setting. They address the changes on different levels. First of all, they provide answers to basic questions on an implementation. They are off-the-box ready-to-run. In a second step, they enable the efficient development of additional follow-up test setups, like functional, integration or scalability testing from the same stock of basic functionality, thus reusing the efforts already invested into the first step.

To achieve this, a test specification language that supports different kinds of testing as well as means for composition, adaptation, modularization and reuse was required. TTCN-3 answers these needs. Its availability is the basis for a broad spreading of the test framework idea. However, TTCN-3 addresses even more user needs; the availability of different presentation formats for different kind of applications. Most noticeable here are the core and the graphical notation. The pool of test patterns, let it be types, data and/or behavior can be developed by test solution providers. Users can then extend the provided functionality on-demand so that their test solutions grow with the users need. A not neglectable property of TTCN-3 is its suitability for different kinds of technologies. Once being familiar with the standardized environment (language, interfaces, execution) efforts can be significantly reduced in order

to develop/use test frameworks for other technologies.

But, TTCN-3 alone has the potential for test frameworks only. Whether this concept will be accepted by the user heavily depends on the availability of practical and efficient tools. As TTCN-3 is a standardized notation there are different vendors supporting this technology. However, for test frameworks to be accepted, tool vendors have to built-in the required support. First concepts of test frameworks have been developed in [9] and implemented in Testing Technologies' tool chain for TTCN-3 [10]. It offers test solutions providers the possibility to design and implement TTCN-3 based test frameworks for a variety of technologies.

References

[1] ETSI ES 201 873 - 1 , v2.2.1, *The Testing and Test Control Notation TTCN-3: Core Language*, Oct. 2002.

[2] ETSI ES 201 873 - 3 , v2.2.1, *The Testing and Test Control Notation TTCN-3: Graphical Presentation Format of TTCN-3 (GFT)*, Oct. 2002.

[3] ETSI ES 201 873 - 5 , v1.0, *The TTCN-3 Runtime Interface (TRI); Concepts and Definition of the TRI*, Oct. 2002.

[4] ETSI DES 201 873 - 6 , v1.0, *The TTCN-3 Control Interfaces (TCI); Concepts and Definition of the TCI*, Mar. 2003, Draft.

[5] S. Schulz, T. Vassiliou-Gioles, *Implementation of TTCN-3 Test Systems using the TRI*. IFIP 14th International Conference on Testing of Communicating Systems (TestCom 2002), Berlin (Germany), March 2002.

[6] I. Schieferdecker, T. Vassiliou-Gioles, *Realizing distributed TTCN-3 test systems with TCI*, IFIP 15th Intern. Conf. on Testing Communicating Systems - TestCom 2003, Cannes, France, May 2003.

[7] J. Grabowski, D. Hogrefe, G. Rethy, I. Schieferdecker, A. Wiles, C. Willcock, *An Introduction into the Testing and Test Control Notation (TTCN-3)*, Computer Networks Journal, Vol. 42, Iss. 3, 2003.

[8] A. Wiles, T. Vassiliou-Gioles, S. Moseley, S. Mueller, *Experiences of Using TTCN-3 for Testing SIP and OSP*, 1st ATS Conference, ACATS Forum Conference, Athens, March 2002.

[9] I. Schieferdecker, B. Stepien, *Automated Testing of XML/SOAP based Web Services*, 13. Fachkonferenz der Gesellschaft fuer Informatik (GI) Fachgruppe "Kommunikation in verteilten Systemen" (KiVS), Leipzig, 26.-28. Febr. 2003.

[10] Testing Technologies' TTCN-3 Tools, *The TT Tool Series*, `http://www.testingtech.de/products`

[11] Tektronix Test Device, *K1297-G20*, `http://www.tek.com/site/ps/0,,2F-15849-INTRO_EN,00.html`