GMD Report 153

GMD –
Forschungszentrum
Informationstechnik
GmbH

Marion Schünemann
Ina Schieferdecker, Axel Rennoch
Mang Li, Claude Desroches

Improving Test Software using TTCN-3

2001

GMD – Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Germany
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

Anschriften der Verfasser/Addresses of the authors:
Marion Schünemann
Dr. Ina Schieferdecker
Axel Rennoch
Institut für Offene Kommunikationssysteme (FOKUS)
Kaiserin-Augusta-Allee 31
D-10589 Berlin
E-Mail: {schuenemann, schieferdecker, rennoch}@fokus.fhg.de

Mang Li
Ludwig-Maximilians-Universität München
Institut für Informatik
Oettingenstraße 67
D-80538 München
E-mail: mang.li@informatik.uni-muenchen.de

Claude Desroches
Conformance Technologies Ltd.
Penetanguishene, Ontario
Canada
E-mail: cdesroche@aol.com

ISSN 1435-2702

Abstract: This paper presents our findings with the Testing and Test Control Notation (TTCN-3). Using recently developed tools, we have successfully implemented a TTCN-3 test suite for the GIOP/IIOP standardized CORBA interoperability protocol. Of particular interest are TTCN-3 improvements over its predecessor Tree and Tabular Combined Notation (TTCN-2). This paper describes the GIOP test suite, its design and implementation and compares it to a TTCN-2 test suite for GIOP. With our comparison, we specifically address test developers with technical knowledge on at least one of the TTCN versions.

Keywords: Test Software, TTCN-3, Abstract Test Specification, CORBA, Conformance Testing, Test Runtime Interface

Zusammenfassung: In diesem Beitrag berichten wir über unsere Erfahrungen mit der Testbeschreibungssprache TTCN-3 (Testing and Test Control Notation). Durch den Einsatz von neu entwickelten Werkzeugen konnte eine TTCN-3 Test Suite für GIOP/IIOP, dem standardisierten CORBA Interoperabilitätsprotokoll, erfolgreich implementiert werden. Die Verbesserungen von TTCN-3 gegenüber seinem Vorgänger, der Tree and Tabular Combined Notation (TTCN-2), sind hier von besonderem Interesse. In diesem Beitrag wird die GIOP Test Suite beschrieben, ihr Entwurf und ihre Implementierung erklärt sowie ein Vergleich mit einer TTCN-2 Test Suite für GIOP hergestellt. Mit unserem Vergleich sprechen wir insbesondere diejenigen Testentwickler an, die bereits technische Erfahrung haben mit wenigstens einer der TTCN Versionen.

Schlagworte: Test Software, TTCN-3, Abstrakte Test Spezifikation, CORBA, Konformitätstesten, Test Runtime Interface

Contents

1 Introduction

Many modern software is based on object-oriented and distributed concepts. CORBA (Common Object Request Broker Architecture) [2] is a widely-used middleware technology with the aim to connect distributed object-oriented software components via so-called ORBs (Object Request Brokers). It is a vendor-independent and programming-language-neutral industrial standard which is available for most computer platforms and operating systems.

One important part of the standard is the General Inter-ORB Protocol (GIOP), which provides interoperability between CORBA implementations of different vendors. The most common and also mandatory mapping of GIOP message transfer is IIOP (Internet Inter-ORB Protocol), which utilises TCP/IP (Transmission Control Protocol/Internet Protocol) connections. To clarify the relationship between GIOP, IIOP, and TCP/IP, it is helpful to view them in the context of protocol layers according to the OSI (Open Systems Interconnection) reference model. Although the mapping of TCP/IP and GIOP/IIOP to the OSI reference model is not exact, one can compare IIOP with session layer protocols and GIOP with presentation layer protocols [16].

To achieve a high probability that a protocol implementation operates with other protocol implementations, conformance testing can be used to test whether the implementation conforms to the standard specification. Conformance testing cannot, however, guarantee interoperability. If failures are detected, conformance testing helps to find out which errors have occurred.

Conformance testing of GIOP/IIOP implementations is the more effective way than pure interoperability testing of CORBA applications as interoperability testing cannot detect cases where protocol entities agree or tolerate the same fault. However, conformance testing and interoperability testing complement each other. The DOPG (Distributed Object Promotion Group) provides a freely available test program, called CORBA/IIOP Interoperability Test Kit [17] for GIOP/IIOP interoperability tests. A conformance test suite for GIOP/IIOP has already been developed using TTCN-2 in the CORVAL2 (CORBA Validation 2) project [5] as part of the VSOrb Validation Suite for CORBA v2.3/v2.4 ORBs. The VSOrb test suite is used by The Open Group to brand CORBA products [19]. The test approach presented here follows the principles of VSOrb, however, the new tests defined in this work allow greater modularity and flexibility.

A commonly used test notation for conformance testing is the Tree and Tabular Combined Notation (TTCN). TTCN has been utilised to specify test suites for many telecommunication protocols, such as ISDN and ATM protocols. The test suites are applied by independent test laboratories to test protocol conformance and to certify the products. Regarding the 2nd edition of TTCN and previous versions, test suites can only be edited with special graphical editors.

TTCN-3 is the current version of TTCN and has recently been standardized by ETSI (European Telecommunication Standards Institute) and by ITU (International

Telecommunication Union). It is a complete redesign and widens the TTCN application area to different kinds of testing applied to different technologies in the telecommunication and IT domain in general. The syntax of the textual TTCN-3 core notation is like a programming language and similar to C++ or Java. No graphical editors are required for the core notation, but could be used if the graphical format of TTCN-3 is applied instead. Because TTCN-3 is a totally new technique, it requires new tool support. First parsers, compilers, run-time environments and editors are available [13] and have been used for this work.

The focus of this work is on testing the conformance of GIOP implementation with respect to the standard specification by using TTCN-3 to develop an abstract test suite. TTCN-3 tools are applied to implement an executable test suite and to run tests. The idea is to show that TTCN-3 is at least or even more suitable for protocol conformance testing than TTCN-2. The test suite will be applied in a further project with the intention to develop open source conformance tests for CORBA products [1].

TTCN-3 is described in Chapter 2, especially the core language and the test suite structure are elaborated. The runtime requirements for test execution are depicted. Chapter 3 gives a short introduction to the CORBA GIOP/IIOP, and an overview on other applications of TTCN-3. The technical details on the advantages of TTCN-3 over TTCN-2 are discussed in Chapter 4. This chapter specifically address test developers who have already some experiences with either TTCN-2 or TTCN-3. Some tools are introduced in Chapter 5. A summary and an outlook are given in Chapter 6.

2 The testing technique TTCN-3

The Tree and Tabular Combined Notation (TTCN) is a standardized test notation. It was first published as a standard by the International Telecommunication Union - Telecommunications Standardization Sector (ITU-T) and the International Organization for Standardization (ISO) in 1992. TTCN is part of the ISO standard 9646 Information technology - OSI Conformance Testing Methodology and Framework (CTMF) [15]. Conformance testing means to test if an implementation under test (IUT) is conformant against the standard specification. It involves testing both the capabilities and the behaviour of an implementation. The purpose of conformance testing is to increase the probability that different OSI implementations are able to interwork. TTCN is used to specify an abstract test suite (ATS). An ATS consists of a number of test cases. Each test case should be designed to achieve one of the specified test purposes. For the execution of test cases, tool support is useful to compile the ATS. The currently widely used version of TTCN is the 2nd edition referred to as TTCN-2 (part 3 of [15]). TTCN-2 is well supported by tools for editing, compiling and executing test suites. In TTCN-2 it is common to use its tabular form to develop a test suite.

The successor TTCN-3 [10] - the Testing and Test Control Notation - has recently been standardized by the European Telecommunication Standards Institute (ETSI). At ITU-T the new standard has been approved as prepublished recommendation (Z.140 and Z.141). Being

the successor of TTCN-2, TTCN-3 inherits all methods and concepts that made TTCN-2 a well-established and wide-spread test notation. The development of TTCN-3 was forced by key players of the telecommunication industries and science to get a single test notation for nearly all the testing needs. Especially the newly introduced support of synchronous communication, i.e. communication based on procedure or operation calls, enables new applications of TTCN while keeping the mature and stable test concepts.

TTCN-3 is a complete redesign of the entire test specification language TTCN. The intention is to modernize TTCN and to widen its application area beyond pure OSI conformance testing. TTCN-3 is not restricted to conformance testing, it also can be used for other kinds of testing, e.g. performance, interoperability, system, and robustness testing. And it can be used for protocol testing (including mobile and IP protocols), service testing (including supplementary services), module testing, and API testing. TTCN-3 is also suitable for testing CORBA infrastructures and CORBA-based systems and applications.

TTCN-3 has a textual format with a syntax like a modern programming language. Main features of TTCN-3 are

- Dynamic, concurrent test configurations to enable scalable, adapting and distributed test setups
- Synchronous and asynchronous communication mechanisms to support the testing of message-based systems (e.g. protocols) and procedure-based systems (e.g. client-server systems)
- Data and signature templates with strong matching mechanisms to support the flexible definition of test data in terms of stimuli to the system under test and observed reactions thereof
- Any type and value parameterization to enable the use of type and behaviour definitions in various contexts
- Assignment and handling of test verdicts to define precisely the evaluation of the system under test as being passed
- Combined use of TTCN-3 with ASN.1

These features and the well-defined syntax and semantics make TTCN-3 a widely applicable and customizable test specification and implementation language. In addition to the textual core notation, TTCN-3 provides other presentation formats, in particular a graphical one based on sequence charts and a tabular one based on tables, to increase the user-friendliness, readability and documentation features of TTCN-3. Using e.g. the graphical presentation format, complex test scenarios can be easily visualized without losing the overview.

The top-level unit of a TTCN-3 test suite is the module, which can import definitions from other modules. A module consists of a declarations part and a control part. The declarations part of a module covers definitions, e.g., for test components, their communication interfaces (so called ports), type definitions, test data templates, functions, and test cases. Figure 1 gives a survey of a TTCN-3 module structure. Some of the basic language elements are described below.

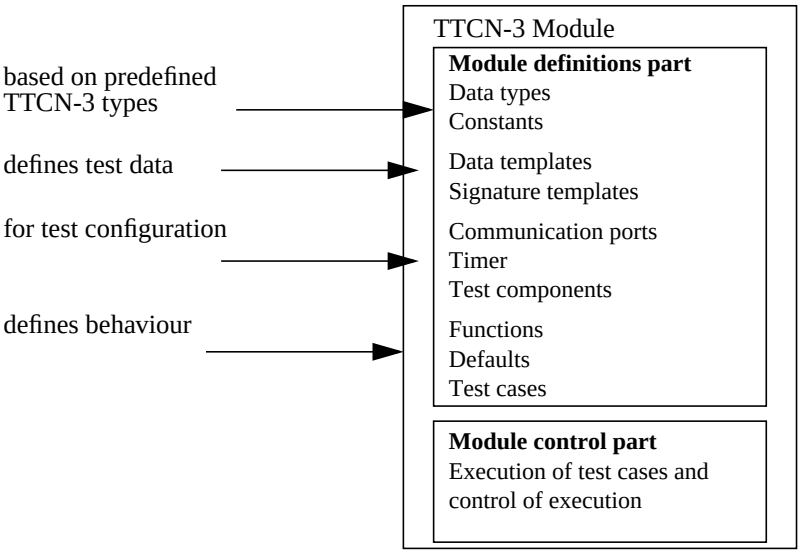

Figure 1: TTCN-3 module

The control part of a module calls the test cases and describes the test campaign. For this, control statements similar to statements in other programming languages (e.g., if-then-else and while loops) are supported. They can be used to specify the selection and execution order of individual test cases. Groups can be used to structure the declarations of a module to ease the readability. Types for messages and signatures are defined as structured types, which can be constructed as record, set, enumerated types, etc. from basic types or other structured types. TTCN-3 supports a number of predefined basic types, which include programming-typical basic types such as integer, boolean and string types, as well as testing-specific types such as verdict type, port and component type. The latter two are used to define the configuration of a test system. Test data to be sent or received via ports are defined as templates. Templates support matching mechanism for the denotation of a variety of expected data.

Test cases describe the probes during the test campaign, i.e., they specify the test behaviour. Test cases are executed by a main test component, which is automatically created when a test case is started. Further parallel test components can be created and terminated during the execution of a test case. The test components (both the main test component and the parallel test components) are connected via well-defined communication interfaces, so-called ports, to the system under test or to other test components. The communication paths between ports can dynamically connected and disconnected as well as be started and stopped. The set of interfaces of the system under test is the border of the test system. By default, it is assumed to be identical to the interfaces of the main test component. If not, it has to be explicitly defined. The structure of a typical component-based test system in TTCN-3 is depicted in Figure 2.

One can express a variety of test relevant behaviour within a test case such as the alternative reception of communication events, their interleaving and default behaviour to cover, e.g., unexpected reactions from the tested systems. In addition to the automatic test verdict

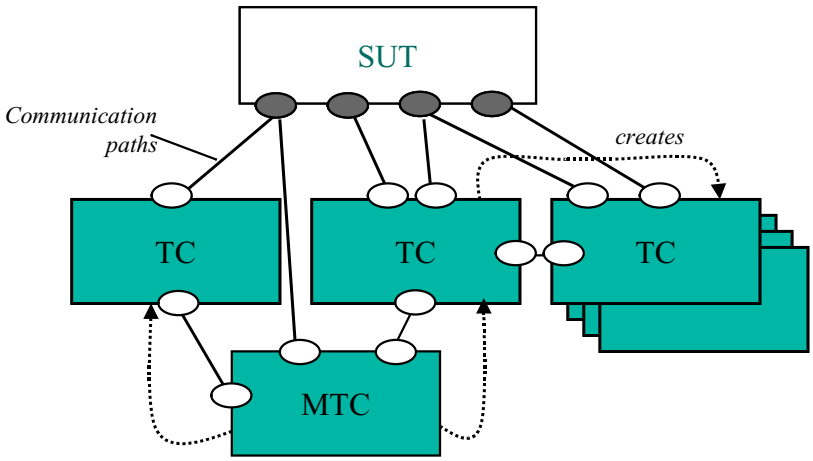

Figure 2: Test components in a TTCN-3 test system

assignment, more powerful logging mechanisms, e.g., for a detailed tracing, are provided.

The import mechanism in TTCN-3 enables the fine granular reuse of almost all TTCN-3 objects. Hence, new test scenarios can be developed on the basis of already available tests. New test scenarios can be for example to reuse functional test cases in the context of integration tests, to reuse integration tests for scalability tests, or to reuse the data and type definition to design new tests.

3 The test suite for GIOP

The General Inter-ORB Protocol (GIOP) is defined to provide interoperability between different ORB implementations. The specification of GIOP consists of three core elements:

- The Message Transport: transport assumptions are made on the used transport layer. The GIOP specification gives a description of how to manage a connection.
- The GIOP Message Formats of the exchanged messages, consisting of message headers and message bodies.
- The Common Data Representation (CDR), which is defined for the "on-the-wire" transfer between ORBs.

Testing the conformance of a GIOP implementation includes to test the following parts:

- the right GIOP message format and ordering
- the mapping of IDL data types to the right CDR
- the behaviour of the implementation in case of errors, e.g. inappropriate message, wrong message content, or erroneous encoding

For each part a set of test purposes must be defined. These test purposes are described as

follows.

Message format and ordering: The test purpose is to verify that after receiving a Request message, the IUT answers with a Reply message which format conforms to the GIOP specification. This includes to verify that the request id is set properly. A similar test purpose is to verify that the IUT sends a LocateReply message that conforms to the GIOP specification, after receiving a LocateRequest message. A third test purpose is to verify that after receiving a Request message followed by a CancelRequest message, the IUT sends no Reply message within a certain period. Because the message format depends on the GIOP version, the format has to be tested for all three versions in extra test cases. Considering the byte ordering, each test case must be realized using little-endian and big-endian.

IDL data type mapping to CDR: The test purpose is to verify that after receiving a Request message, invoking for example the operation return long, the IUT answers with a Reply message which format conforms to the GIOP specification and the return value of type long is in the right CDR. This has to be tested for all GIOP versions, for all IDL data types, and for both byte orders.

Behaviour in case of errors: The test purpose is to verify that after receiving a Request message containing a wrong magic value, the IUT answers with a MessageError message. This also can be tested for a Request message containing a non specified message type or a non existing version number.

Different test configurations have been proposed and evaluated in [8]. Here we will restrict to depict the chosen configuration:

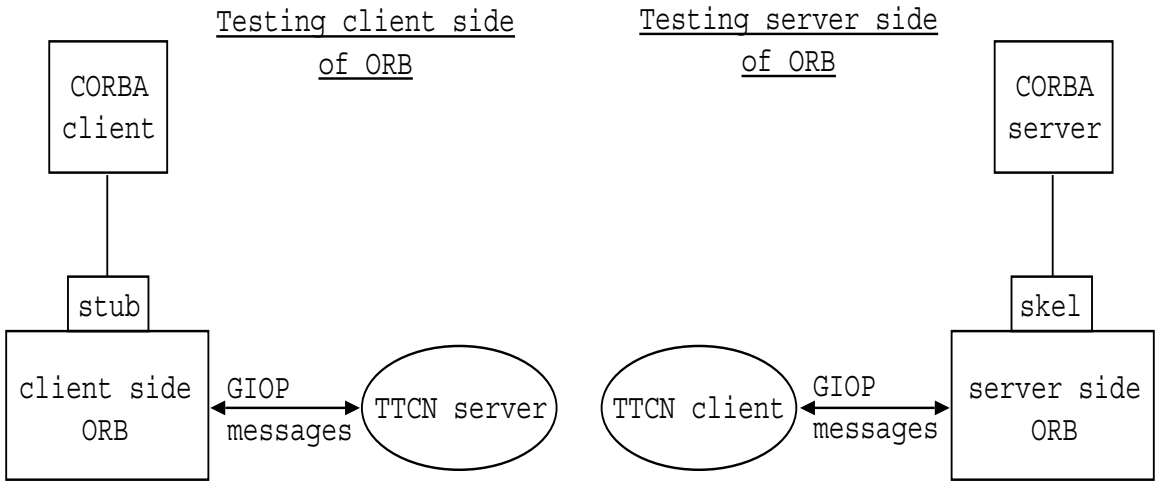

Figure 3: Chosen test configuration

In the chosen test approach, two abstract test suites must be specified, one to test the client side of the ORB and another to test the server side of the ORB. The decision for this approach is

based on the following four reasons:

- The approach of conformance testing is used to avoid many-to-many ORB interoperability testing. It is assumed that conformant implementations guarantee a higher probability of interoperability between ORBs.
- Using CORBA components at the operation-based interfaces and TTCN components at the message-based interfaces, control over the test is given on both sides because of the two test suites.
- In this work a TTCN-3-to-Java compiler is used to get an executable test suite. For testing ORB implementations which are not written in Java, a reimplementation of the CORBA client and CORBA server or an adaptation of an existing CORBA client and CORBA server in another programming language is relatively simple. TTCN components in this case would assume a TTCN-3 compiler, which generates code in a programming language this ORB supports.
- TTCN-3 supports operation-based interfaces, which is a new feature of TTCN. But no gateway between TTCN-3 and CORBA is available at this point in time.

The server side of the ORB has been considered in this work. Testing the client side is analogous. A TTCN-3 ATS has been specified. Using a TTCN-3-to-Java compiler and a runtime environment, an ETS has been developed to test the server side of the ORB.

Here we also like to mention, that some experiences on the application of TTCN-3 test suites have been made and reported in [6] and [9]. Due to its availability as an executable test suite in both versions, TTCN-2 and TTCN-3, the GIOP test suite appears to be a good candidate for a detailed technical comparison of the language features. This comparison is given in the following section.

4 Comparing TTCN-3 to TTCN-2

Having developed a test suite for the GIOP protocol in TTCN-3, TTCN-3 met the expectation of being directly and easily applicable for testing of message based systems. In particular, TTCN-3 showed several distinct advantages over TTCN-2. In another study, TTCN-3 has also been shown to be effective for the testing of operation-based interfaces, components and applications [9].

The advantages of using TTCN-3 for defining a GIOP test suite have been firstly in the easy use of the textual core notation of TTCN-3. For editing the abstract test suite, the GNU Emacs with a special TTCN-3 mode [18] has been used. That helped to develope the ATS in an integrated manner with traversing, syntax and semantics checking, and code generation capabilities efficiently. In addition, the test specification can be developed and is readable in every text editor without having the need for cost-intensive development environments.

The syntax of TTCN-3 showed to be suitable to specify GIOP messages relatively fast and well readable. Parameterisation helped to reuse type and template definitions in a succinct manner.

In the ongoing work, tests for the client side of ORBs will be developed. With the module and import concepts of TTCN-3, significant parts of the existing test suite can be reused to specify and implement the additional tests. Even, the two test suites can be combined into one so that the test suite user has to perform only one set of test cases. Technically however, two different types of main test components will be used in that integrated test suite to reflect the different configurations of client and server side tests. An integrated test suite will be more practical and comfortable. Even, the setup of a test campaign of a given ORB under test will be simplified.

Another aspect is the start of the CORBA client which is needed to perform the tests. By means of two external functions to start the CORBA client and to stop it after test execution, which are used in the control part, the tests can control that CORBA client automatically.

Beyond the concrete technical advantages of TTCN-3 for the GIOP tests, there are further ones. In practice TTCN-3, requires less code than TTCN-2 to express the same test behaviour. The absence of information related to the test suite structure and indexes for test cases, test steps and defaults reduce the size of a TTCN-3 test suite considerably. This information can be provided, but its presence is tool dependent.

Expressing test behaviour in TTCN-3 is more compact. This has the further benefit of making code easier to read. Sequential test behaviour in TTCN-3 is expressed by separating individual statements with a semicolon ';'. TTCN-2 expresses sequential test behaviour by a change in the level of indentation, while alternative behaviour is on the same level of indentation. In fact, if two alternatives show the same subsequent test behaviour, this has to be denoted in TTCN-2 twice while it has to be denoted in TTCN-3 only once (see Figure 4 for a comparison).

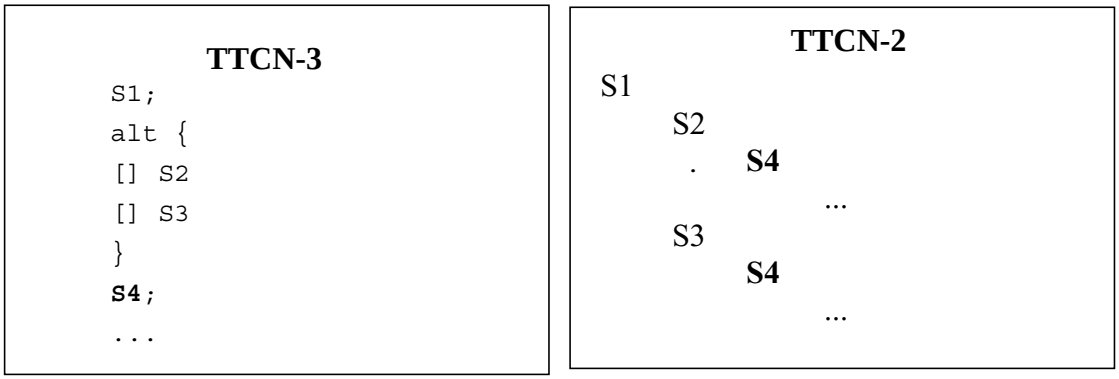

Figure 4: Expressing sequential and alternative behaviour TTCN-3 vs. TTCN-2

In TTCN-2, readability is seriously affected because of the change in level of indentation. Less and less space is available, both horizontally and vertically for expressing new sequential statements. Code necessarily gets spread across several pages when printed, and requires the

developer to scroll up and down to view code.

Another advantage relates to dynamic test configuration. In TTCN-3, communication paths can be dynamically established and released. Test component and test system test ports can be dynamically mapped to and unmapped from one another. Furthermore, test component test ports can be connected to and disconnected from other test components test ports. One may also define one-to-many connections, that is, a single test port may be connected to several other test ports at the same time. As a further improvement over TTCN-2, a test port may be connected to one or more test ports at different points in time during the execution of a given test. In contrast, TTCN-2 allows only static predefined test configurations with fixed communication paths. It does not allow broadcasting or loopback type communication. In TTCN-2 communication is limited to exactly two parties: a sender and a receiver. Because TTCN-3 test configurations can be constructed dynamically, it is easier to determine if and when a connection between two test ports was established and released. This offers further opportunities to identify sources of potential problems in a system under test.

Another welcome simplification is the replacement of the unfamiliar TTCN-2 terms Points of Control and Observation (PCOs) used to indicate a communication between a test component and the IUT, and the term Coordination Points (CPs) used to identify communication between two test components with the more familiar and more intuitive term ’test port’. This change in terminology is a result of decoupling TTCN-3 from OSI and conformance testing specific terminology.

In addition we benefited from the following TTCN-3 features:

- Test campaigns were directly specified within the TTCN-3 control part. No supplementary test management tool is required (e.g. to control metering programs).
- The platform and SUT adapter interfaces as specified in the TRI [11], defined using IDL. TRI provides a clearer separation of design issues when compared with GCI [14]. TRI will be standardized. This ensures that the ATS will run as is, without need to modifications, due to SUT adaptation implementations differences.
- The absence of global variables promotes ease of use and code re-usability. Furthermore unwelcome side effects are avoided.

There are additional advantages, which we like to bring to the software test community attention:

- Experiences has shown that many software engineers unfamiliar with the OSI reference model have difficulties getting used to the OSI terminology (e.g. PDU, ASP etc.). TTCN-3 is no longer closely coupled with the OSI Reference Model and its inherited terminology and therefore it is easier to read for programmers familiar with text based programming languages (such as C, C++, ADA, etc.). Consequently, TTCN-3 is expected to become the most widely used standardized test language.

- The language could be extended by other attributes to objects in a test module, by use of external functions as well as by the inclusion of external data. For example, there exists a mapping from IDL to TTCN-3 to make direct use from user defined IDL definitions. This enables an efficient test development for software components developed with IDL.
- In contrast to TTCN-2 the new standard covers ASN.1 data types more complete, which is e.g. essential to UMTS protocols and services.
- TTCN-3 features an interleave construct which greatly reduces the amount of code that developers must provide to completely specify test cases (cp. section 20.4 of [10]).
- TTCN-3 provides users with the ability to extend existing different presentation formats or to develop their own presentation format.
- Regular expressions were added and improve the matching mechanism for use in templates.

5 Tool support

TTCN-3 has already tool support in terms of editors, syntax and semantics checkers, compilers and runtime environments.

As TTCN-3 defines test cases on an abstract level, test developers can focus on the development of tests, instead of wasting time on how to implement them on a given platform or operating system. The generation of executable test suites requires a TTCN-3 compiler (or interpreter) and the execution of tests requires a runtime environment together with an appropriate adaptation. For a given platform and technology, an adaptation of the generated code can typically be reused. A set of tools from the company Testing Technologies [13] is available for the compilation and execution of TTCN-3 tests. It is used in this work and described in the following.

The TTCN-3-to-Java compiler TTthree is used to compile the ATS. The test adapter that among other things, performs the TCP/IP communication on the test device uses the TTCN-3 Runtime Interface (TRI). TRI is an interface, e.g. to mapped ports and to timer implementations. It will become an ETSI standard by the end of this year. The TTCN-3 Control Interface (TCI) is an interface to the test management, to intercomponent communication and to type and data implementations. TCI is not yet standardized by ETSI, but is currently under development. To run the tests the test management tool TTman and the TTCN-3 run-time environment TTrun are used.

In the following, a generic TTCN-3 test system implementation is described. Figure 5 shows the structure of a TTCN-3 test system, which illustrates the connection between TRI, TCI, the system under test (SUT) which resides the IUT, and the test system user.

A test system implementation must take the following steps:

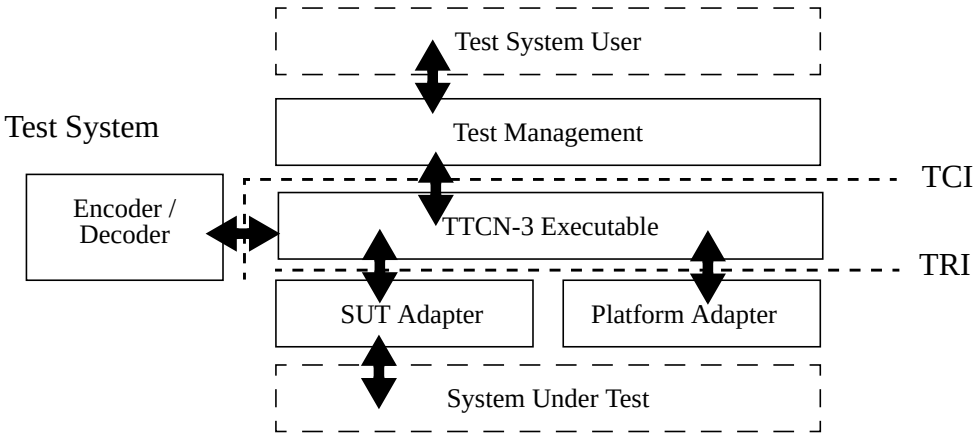

Figure 5: TTCN-3 test system

- The compilation of the TTCN-3 test cases to executable test code
- The adaptation of the test code to the system under test by realizing the communication to the system under test via its interfaces, the implementation of external functions, and the handling of timers.
- The treatment of test data by means of encoding and decoding capabilities

The test system user interacts with the test system via the test management. The TTCN-3 executable (TE) is responsible for the execution of the TTCN-3 test cases as defined in the control part. The TE is connected to the test management via TCI and to the SUT adapter and the platform adapter via TRI. Test components are created and removed by the TE. The TE is responsible for the encoding and decoding of test data. The TE gives instructions to the SUT adapter as well as to the platform adapter, e.g relating to the sending of messages to the SUT, to the execution of external functions, or to timer operations. Verdicts as the result of executing a test case are offered via the test management to the test user.

The Test Management part realizes the overall management of the test system. It is connected to the user of the test system via an interface. The test management is responsible for the proper invocation of TTCN-3 modules. If necessary, it passes module parameters to the TE. It collects and maintains the test results and logging information from the TE.

In the SUT Adapter, the message and procedure based communication of the TTCN-3 test system with the SUT is adapted to the particular execution platform of the test system. The SUT adapter implements the real test system interface. It realizes the mapping of the ports defined in the ATS to the real test system interface.

The platform adapter implements external functions and timer operations as defined in the TTCN-3 module. Timers can be started, stopped and read by the TE via an interface. The platform adapter informs the TE of expired timers.

6 Conclusions

This work dealt with two main topics: (1) conformance testing to promote interoperability of protocol implementations in a distributed environment and (2) the collection of practical experiences with the new test notation TTCN-3 applied to protocol testing. The tests in the abstract test suite can be used to validate server side interworking protocols of CORBA ORBs. A test adapter has been developed to enable their execution.

Good experiences could be made with the new test notation TTCN-3 in the context of protocol testing. The development of the test specification in TTCN-3 was fast and efficient. The TTCN-3 tools could be used to implement an executable test suite and to run the tests. However, a predefined set of methods would be preferable for the encoding of TTCN-3 types to Java types, e.g to map the TTCN-3 integer to the Java integer or a TTCN-3 record to a Java byte array.

Advantages of TTCN-3 test suites as opposed to TTCN-2 are that they are readable in every text editor and do not require expensive development environments. The Java implementation of the tests generated by the TTCN-3-to-Java compiler can easily be adapted to different operating systems.

The test suite developed in this work will be a contribution to the COST (CORBA Open Source Testing) project [1]. The aim of this project is to establish and maintain CORBA tests using an open source process. Users and providers of CORBA technology can use and discuss those tests to analyse their products. The project is intended to complement formal branding programs such as the one administered by The Open Group. The development of the test suite is an ongoing effort due to its inclusion in COST.

In the context of CORBA conformance testing, we see the following advantages due to the introduction of TTCN-3 [7]:

- The user acceptance of TTCN-3 is expected to be higher than TTCN-2 due to its wider scope of application (e.g. for synchronous API testing), e.g. TTCN-3 has been identified to cover all CORBA conformance aspects.
- The distribution of TTCN-3 documents is expected to be larger due to its flexibility of the presentation formats, e.g. programming code styles or graphical sequence chart format in addition to the classical tabular and tree format.
- Within the OMG working group on an UML profile for testing the TTCN-3 concepts will be related to the UML metamodel to allow the integration of test development and system development techniques and to have a direct bridge to testing software systems by means of TTCN-3.

This paper also aims to provide an input contribution to the discussion on strategies for industrial companies who have to decided to migrate from TTCN-2 to TTCN-3 [12].

Automated translation of TTCN-2 test suites together with powerful tooling support for TTCN-3 helps in that transition. Due to the ongoing maintenance and support of TTCN-3 by ETSI, standardization activities will continue and will increase the benefits for TTCN-3 users. For example, the test control interface TCI will be standardized enabling the easy distribution of test components in a distributed network as well as the integration of encoders/decoders into a TTCN-3-based test system.

7 References

[1] CORBA Open Source Testing: http://cost.omg.org

[2] Object Management Group, Inc (OMG): The Common Object Request Broker: Architecture and Specification, Revision 2.3, June 1999.

[3] M. Ebner: A mapping of OMG IDL toTTCN-3, Technical Report SIIM-TR-A-01011, Unniversiy of Lübeck, July 2001.

[4] J. Grabowski: TTCN-3 - A new Test Specification Language for Black-Box Testing of Distributed Systems, June 2000.

[5] M. Li et al.: Experience report on Conformance Tests for CORBA ORBs, 2nd Asia-Pacific Conference on Quality Software, Hong Kong, Dec.2001

[6] I. Schieferdecker et al.: Systematic Testing of Internet Protocols, Africom 2001.

[7] I. Schieferdecker et al.: Testing of CORBA products, QWE'2002, March 2002.

[8] M. Schünemann: Development of a TTCN-3 based test suite for CORBA GIOP/IIOP. Studienarbeit, Technical University of Berlin, Fac. of Electrical Engineering and Computer Science, Berlin, Dec. 2001.

[9] A. Yin: Operation based testing on different abstraction levels, Paris, Dec. 2001.

[10] ETSI & ITU-T: The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. October 2001.

[11] ETSI: The TTCN-3 Runtime Interface (TRI). October 2001.

[12] Open TTCN, Finland: www.openttcn.com

[13] Testing Technologies, Germany: www.testingtech.de

[14] INTOOL project: Generic Compiler Interpreter (GCI) Interface.

[15] International Organization for Standardization (ISO), Information Technology, Open Systems Interconnection, Conformance Testing Methodology and Framework (ISO/IEC 9646), Geneve, November 1998.

[16] W. Ruh et al.: IIOP Complete Understanding CORBA and Middleware Interoperability, 1999.

[17] DOPG, Japan: http://www.dopg.gr.jp/iiop/iiop_e.html.

[18] Debian: http://packages.debian.org/unstable/editors/ttcn-el.html.

[19] The Open Group branding program on CORBA validation: http://www.opengroup.org/corval2