# Testing the TINA Retailer Reference Point

Mang Li, Ina Schieferdecker and Axel Rennoch

GMD FOKUS
Competence Center for Testing, Interoperability and Performance
Kaiserin-Augusta-Allee 31, D-10589 Berlin
phone: +49 30 3463-7000, fax: +49 30 3463-8000
www.fokus.gmd.de/tip

**Abstract**

*This paper shows that testing open object-based distributed systems is, in principle, feasible by applying the ISO/ITU-T Conformance Testing Methodology framework. In this context, a gap in the (semi-) automatic test development and execution process has been identified and bridged. We have developed an approach to the application of the standardized test notation TTCN for testing ODL-based implementations. The equivalents of TTCN features in formal TINA specifications have been identified. Further, a TTCN-based test system implementation has been integrated in the CORBA environment. Test specification and execution are discussed w.r.t. the TINA Retailer Reference Point.*

## 1. Introduction

Distributed processing is important for the realization of telecommunication services in today's heterogeneous information networking environment. The *reference model of open distributed processing* (RM-ODP) [OD95] is a framework developed by ISO and ITU-T to provide concepts and guidance for the design and programming of distributed systems. The five viewpoints, the object model and the distribution transparency belong to the basic concepts of RM-ODP. RM-ODP describes also principles for the conformance assessment of ODP specifications and implementations.

Part of the RM-ODP concepts is specialized in the *common object request broker architecture* (CORBA) [CO98] which is standardized by the industry consortium OMG (object management group). CORBA provides an infrastructure for distributed object systems. The *interface definition language* (IDL) of OMG supports the formal specification of object interfaces.

CORBA is used for the realization of most of the TINA (*telecommunications information networking architecture*) services. The TINA business model defines business roles and business relationships as such: a consumer uses services that are provided by a TINA system, a broker supplies information that enables a stakeholder to find other stakeholders and services, a retailer serves stakeholders by offering them access to services, a third party provider supports retailers or other third party providers with services, and, a connectivity provider manages the communication network.

The TINA *object definition language* (ODL) is an extension of the OMG IDL for the computational specification of services. The TINA reference point concept [RP96] adopts the conformance assessment concept of RM-ODP. That is, conformance deals with the specification-implementation relationship; the method for the determination of this relationship is conformance testing; and the conformance requirements are defined by reference points. Reference points in TINA systems reside on boundaries of objects, whereas an object can be of different granularity. It can be, for example, a computational object, a service component, a business role or a business domain in multiple roles. In fact, neither RM-ODP nor TINA defines precisely the method for conformance testing.

Testing of such distributed object systems has been investigated only recently. [LC97] and [UA97] present experiences in using CORBA-based test environment. The results obtained in this area are still insufficient. To our best knowledge, the use of formal specification-based testing for distributed systems has not been examined yet. Although, the necessity has already been recognized.

On the other side, in the area of conformance testing for OSI protocol-based real systems, a methodology has already been established [CT97]. The central concept of this framework addresses the abstract specification of conformance tests. For this purpose, the test notation TTCN (*tree and tabular combined notation*) is defined. The well-

defined syntax and semantics of TTCN (see e.g. [KW98]) support unambiguous behavior description of test systems and translation from TTCN test suites into program code. TTCN is mostly used for testing protocol implementations above the data link layer.

Our work is to evaluate the usability of TTCN for testing object-oriented systems based on the general concepts of RM-ODP. Currently, we focus on the conformance testing of object systems that are specified using TINA ODL and realized on top of CORBA. In this paper, our approach is presented with an example test suite for the TINA *retailer reference point* (Ret-RP). A related scenario of the Ret-RP is introduced in *section 2*.

The approach covers both the abstract specification and the execution of tests. The development of abstract specification is discussed in *section 3* and *4*. The test notation TTCN is used for the test specification. For this, a set of mapping rules are defined, which is denoted by *ODLtoTTCN*. The mapping rules support translation of ODL definitions into TTCN declarations, whereas standard TTCN constructs are used. ODLtoTTCN can be used by manual or semi-automatic test development. The proposed method is applied to the specification of an *abstract test suite* (ATS) with selected test cases for the Ret-RP, which is explained in *section 5*.

The test execution aspect of the approach is discussed in *section 6*. A gateway component is introduced, which provides the general test access to CORBA-based applications. We use the Generic Compiler/Interpreter Interface (GCI) [IN98] for the adaptation of TTCN-based test systems to the interface of the gateway.

The usability of the testing approach is shown by real tests for an independent implementation of the Ret-RP. In *section 7*, we describe the systems which are involved in the tests and present the results of the tests.

*Section 8* with conclusions closes this paper.

## 2. TINA Retailer Reference Point

In TINA, a reference point is defined as the specification of a particular set of conformance requirements. Reference points consist of interfaces that describe potential interactions between various TINA entities. The retailer reference point (Ret-RP) is part of the TINA business model and defines the interactions between stakeholders in the consumer business role and stakeholders in the retailer business role.

The specification of Ret-RP [RT98] is separated into an access part and a usage part. The access part contains interfaces that are required to establish a contractual relationship between a consumer and a retailer. The usage part describes service independent interactions between a consumer and a retailer during service sessions. Conformance to the access part is considered independently from the usage part.

In the following, we present an example scenario given in [RT98]. This scenario describes the most primary interactions between a consumer and a retailer. One of the possible sequences of traces is illustrated in *Figure 1*[1].

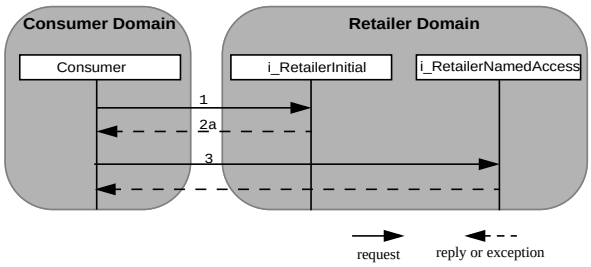


**Figure 1** Example Scenario of the Ret-RP

Step 0: The `i_RetailerInitial` interface represents the initial contact between a consumer and a retailer. A retailer publishes a reference of its `i_RetailerInitial` interface in a way that a consumer can obtain it[2].

Step 1: When a consumer has the initial reference, it calls the `requestNamedAccess` operation on `i_RetailerInitial` as a named user[3], with his/her authentication information passed as parameter.

Step 2: If CORBA security services have been used, two outcomes of the operation `requestNamedAccess` are possible:

2a) If the authentication information is valid, the operation returns successfully with a reference to an `i_RetailerNamedAccess` interface, which indicates that an access session has been established.

2b) If the authentication information is invalid, `e_UserPropertiesError` exception is raised, which indicates that an access session could not be successfully established.

Step 3: Following the case 2a), with the reference of `i_RetailerNamedAccess`, the consumer can inform the retailer about himself (`setUserCtxt`), discover services (`discoverServices`), start a service session (`startService`),..., or terminate the access session (`endAccessSession`).

1. The event traces are similar to MSCs, but not standardized.
2. Usually, a retailer registers it's initial reference with the Naming Service. It may also store the reference in a file, to which a consumer has access.
3. The case of anonymous user is not considered in this scenario.

## 3. ATS Development

The ATS development for testing the conformance of TINA systems is based on the TINA reference point specifications. A reference point specification, such as [RT98], consists of both non-formal description (in text) and semi-formal description (in ODL). Static information, such as data types, constants and operation signatures, are specified in formal ODL syntax.

The test notation TTCN has been developed to support abstract specification of conformance tests for OSI protocol-based systems in form of abstract test suites (ATSs). An ATS is composed of a test suite overview, a declarations part, a constraints part and a dynamic part. The communication between test systems and SUTs is described by test events, that is, sending and receiving PDUs (*protocol data units*) and/or ASPs (*abstract service primitives*) via PCOs (*points of control and observation*).

Obviously, there are differences between the OSI reference model (OSI-RM) and the object model that TINA systems are based on (see below). In our work, it is desired to use existing standardized syntax of TTCN rather than to introduce new constructs. In order to describe the communication between test systems and real TINA systems in terms of PCOs, ASPs and PDUs, it is to identify the equivalents between elements of the OSI-RM and elements of the TINA object model.

In the TINA object model, two roles are distinct: the client role and the server role. An object in the server role provides service to another object which is in the client role in this relationship. The same object can play the server role in one relationship and the client role in another relationship. As shown in *Table 1*, the client-server relationship is equivalent to the user-provider relationship in OSI-RM.

**Table 1** Equivalents in OSI-RM, TINA and TTCN

| OSI Reference Model | TINA Object Model | TTCN |
|---|---|---|
| Service provider role | Server role | [a]) |
| Service user role | Client role | [a]) |
| Service access point (SAP) | Interface | PCO |
| Service primitive | Operation | ASP |

a. This leads to the introduction of different PCO types (see the mapping for interface declarations below).

As an object provides services at its interfaces, an interface is comparable with a service access point (SAP) in OSI-RM. In the testing of OSI-based systems, PCOs rest mostly at SAPs. Therefore, for testing TINA systems PCOs can be associated with object interfaces. Service primitives in OSI-RM are determined as equivalents to operations[4] provided at object interfaces, because operations define service-related interactions between objects in the client role and objects in the server role. In TTCN, service primitives are described by ASPs. The translation from operations into ASPs is supported by mapping rules that are explained in the next section.

The mapping rules, called ODLtoTTCN, are defined to obtain TTCN declarations from the static part of TINA specifications (see *Figure 2*).

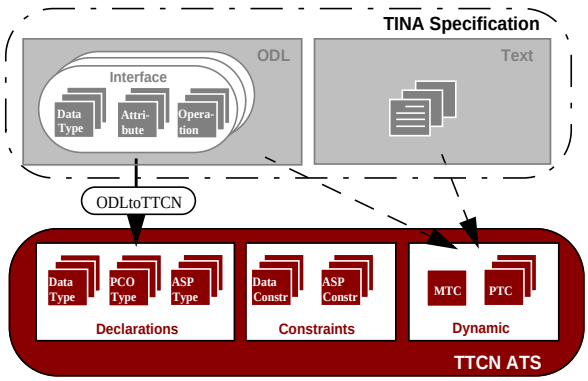


**Figure 2** From TINA specification to TTCN ATS

Besides the translation of ODL types into TTCN types during test preparation, the definition of ODLtoTTCN aims also at the support for generic encoding and decoding of typed values at run-time. Since a TTCN compiler produces language-specific interpretation of TTCN typed values (e.g. in C), as *Figure 3* shows, encoding and decoding deal with conversion of typed values between a TTCN compiler-dependent form and a form that is compliant to an appropriate ODL language binding (e.g. in C++).

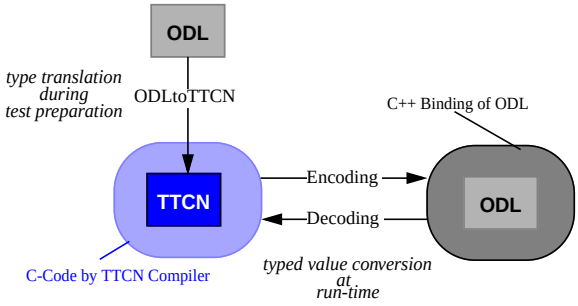


**Figure 3** Translation of type and typed values

The key problem in the encoding and decoding is to determine the type of a value in the target form. TTCN provides fewer basic types than ODL. For example, an integer type in TTCN is `INTEGER`, but can be `short`, `long`, `unsigned short` or `unsigned long` in ODL. Since all IDL integer types are mapped to `INTEGER` in TTCN, the

4. Stream interfaces will be considered in future work.

coding functions must be capable to distinguish them. To support generic coding functions that are common for all ODL types, ODLtoTTCN encloses some naming conventions and help types, which can be considered as guidelines for developing ATSs.

The dynamic aspect of an object system (e.g. ordering of operation invocations, pre- or post-requirement of an operation invocation) is currently covered by textual descriptions, either in the behavior text of ODL specifications or at other places in TINA documents. Formal behavior description is a topic of on-going research [BH97]. Currently, the dynamic part of an ATS must be derived based on the analysis of the textual description in TINA specifications.

## 4. ODLtoTTCN

In this section, an overview of the mapping rules ODLtoTTCN is given. Further details can be found in [LM98].

### Data Types

The mapping for basic data types adopts the ones defined by The Open Group for the translation from IDL to GDMO [OG97], except that the floating-point types are not supported by TTCN.

The mapping for ODL `struct`, `sequence` and `enum` types follows the suggestion in [OG97], too:

`struct` -> `SEQUENCE`,
`sequence` -> `SEQUENCE OF`,
`enum` -> `ENUMERATED`.

The translation for `union`, `array` and `any` types differs from The Open Group's specification (see below).

In [OG97], an ODL `union` type is mapped to an ASN.1 `CHOICE` type. The information on the discriminator is lost. Thus, we use an ASN.1 `SEQUENCE` type. The first field of the `SEQUENCE` contains the discriminator. The second field is a `CHOICE`, with each element of the `CHOICE` associated with a member of the `union` type. To distinguish `union` from `struct` (both are mapped to ASN.1 `SEQUENCE`), all `SEQUENCE` types involved in the ATS that are derived from ODL `union` types are collected in an auxiliary type - a `CHOICE` named `TheUnions`. The identifier of each `CHOICE` element is constructed by the abbreviation `uni_` and the appropriate scoped name of the `union` type.

The ODL `array` types are almost equivalent to the ODL `sequence` types, except that an `array` has a fixed length. Therefore, an ODL `array` type is mapped in an ASN.1 `SEQUENCE SIZE(n) OF` types, whereas `n` is the length of the `array`. This mapping is identical to The Open Group's mapping. In addition to that, to distinguish `array` types from `sequence` types, the auxiliary type `TheArrays` is defined. Similar as `TheUnions`, each element of `TheArrays` is associated with a type derived from an ODL `array` type, with the identifier constructed with `ary_` and the scoped name of the `array` type.

The `any` type can be used to specify a value of any ODL types. The actual type and value contained in an `any` typed value can be determined at run-time. In order to support the generic coding functions, the auxiliary type `TheAnys` is constructed. Each element in `TheAnys` corresponds to a basic ODL type, which may be associated with `any` at run-time. Currently only basic ODL types are used by the mapping for the `any` type. To include structured types, The Open Group's mapping will be considered in further study.

### Exception Declarations

An ODL `exception` declaration has a `struct`-like structure, is therefore mapped to an ASN.1 `SEQUENCE`, like an ODL `struct`.

### Object References

The mapping for object references utilizes the IOR (*interoperable object reference*). An IOR is the global representation of the corresponding object and is composed of ASCII characters. Thus an IOR is mapped to an `IA5String`.

### Interface Declarations

According to *Table 1*, object interfaces are mapped to PCOs. To test the behavior of a SUT, a test system emulates the counterpart of the SUT. In order to describe client-related and server-related test behavior, the following two PCO types are defined:

- `PCOtoSUTServer`: PCOs of this type correspond to interfaces *supported* by the SUT and to be checked by the test system.
- `PCOtoSUTClient`: PCOs of this type correspond to interfaces *required* by the SUT and emulated by the test system.

To support multi-server or multi-client test scenarios, multiple PCOs can be derived from the same interface declaration.

The identifier of a PCO is constructed by the prefix `PCO` that is followed by an arbitrary number, the string "__" (double underscore), and the scoped name of the appropriate interface. For example,
`PCO1__TINARetRetailer__i_RetailerInitial.`

### Operation Declaration

From operation declarations, ASP types are derived (see *Table 1*). Typically, an operation declaration is mapped to the following three ASP types (ASN.1 is used for the ASP type definitions):

- An ASP type for *requests*. The identifier of the type is composed of the scoped name of the corresponding interface, plus a prefix and a suffix. The prefix is `pCALL__` and the suffix consists of "__" and the name of the operation. The body of this ASP type is a `SEQUENCE`. Keeping the order of their definitions in the ODL specification, each element of the `SEQUENCE` is associated with an `in` or `inout` parameter of the operation.
- An ASP type for *replies*. The naming is almost identical to the previous one, except that the prefix of the identifier is `pREPLY__`. The body of the type is also a `SEQUENCE`. The first element is reserved for the return result. Rest of the elements convey `inout` and `out` parameters of the operation, if present. The order of the parameter list is also preserved.
- An ASP type for potential *exceptions*. The naming obeys the same rules as the first one. The prefix of the identifier is `pRAISE__`. Since only one of the potential exceptions is raised once, the ASP type is built upon the `CHOICE` type. Each element of `CHOICE` is associated with a potential exception of the operation, with the identifier constructed by the scoped name of the exception and the prefix `exc_`.

## 5. Ret-RP Test Suite Specification

The mapping rules ODLtoTTCN presented in section 4 are applied to the ODL specification of TINA Ret-RP to serve as the basis to develop an ATS. The declarations part of the ATS consists of permanent declarations (indicated by `p` in the following) and SUT dependent declarations (indicated by `d` in the following). The permanent declarations are common for all CORBA based SUTs.

The declarations part is subdivided into:

- ASN.1 type definitions for:
  - basic CORBA IDL types (p),
  - SUT specific data types (d),
  - CORBA system exceptions (p),
  - SUT specific exceptions (d).
- Test suite parameter, selection expression, constant and variables declarations/definitions (d).
- PCO type declarations for `PCOtoSUTServer` and `PCOtoSUTClient` (p).
- PCO declarations for SUT specific interfaces (d).
- Test suite operation definitions for general purpose operations, for example, to obtain the reference of an initial interface (p).
- ASP type definitions for SUT specific operations (d).

The (data) constraints and dynamic (behavior) parts of the ATS are test purpose related and therefore SUT dependent. Even though, since the target SUTs use the CORBA infrastructure, our first experience shows that the test behavior can be specified in a common style, including the use of test steps and test suite operations for getting initial interface references, the use of timers to prevent deadlocks, as well as the consideration of CORBA system exceptions.

The ATS for the Ret-RP contains selected test cases according to the scenario introduced in section 2. The test case that covers the successful establishment of an access session (*Step 2a* in section 2) will be discussed in section 7, in the context of test execution. We explain in the following a test case that checks the usage of the exception `e_UserPropertiesError` (*Step 2b* in section 2).

The test case presented in *Table 2* characterizes the following test behavior: with the test preamble `GetInitialRef`, a reference of the interface `i_RetailerInitial` is obtained (line 1); a request on the operation `requestNamedAccess` provided by `i_RetailerInitial` is then sent by the tester, whereas the *invalid* authentication information is passed (line 2); the timer `Timer1` is started (line 3); if an exception to `requestNamedAccess` is received by the tester before the timeout of `Timer1`, and the content of the exception matches the appropriate constraint, which is in this case `e_UserPropertiesError`, `Timer1` is stopped and the final verdict `PASS` is given (line 4). In case of timeout of `Timer1`, the verdict `INCONCLUSIVE` is given (line 5). Other alternatives are captured by the default test step `Default_1`. It encloses test events for receiving CORBA system exceptions that lead to the verdict `INCONCLUSIVE`, and `OTHERWISE` pseudo test events which result in the `FAIL` verdict.

## 6. Distributed Test Environment

The SUTs we are considering here are TINA services specified using ODL and implemented on top of the CORBA ORB. Since in the test, a test system emulates the counterpart of the SUT, the external behavior of the test system must be integrated in the environment, where the SUT resides. It means that a bridging between non-CORBA test systems and the CORBA ORB is required.

As introduced previously, the specification of the behavior of a test system is independent from a particular programming language or tool support. In contrary to that, the implementation of such a test system deals with concrete TTCN compilers and CORBA ORB products. In the implementation, it is desired to obtain reusable components that are as much as possible independent from tools. Thus, we separate the code that is generated by TTCN compilers from the program part which provides general bridging between TTCN-based test systems and the

**Table 2** Test case example

| Test Case Dynamic Behaviour | | | | | |
|---|---|---|---|---|---|
| **Test Case Name** : EstablishAS_inv2 | | | | | |
| **Group** : | | | | | |
| **Purpose** : To verify that in the case that CORBA security services are used, on receipt of an invocation on the operation requestNamedAccess() of the interface TINARetRetailerInitial::i_RetailerInitial, with invalid authentication information, an exception TINARetRetailerInitial::e_UserPropertiesError is raised. | | | | | |
| **Configuration** : | | | | | |
| **Default** : Default_1 | | | | | |
| **Comments** : | | | | | |
| **Selection Ref** : SecurityServiceUsed | | | | | |
| **Nr** | **Label** | **Behaviour Description** | **Constraints Ref** | **Verdict** | **Comments** |
| 1 | | +GetInitialRef (PCOName__PCO1__TINARetRetailerInitial__i_RetailerInitial, ObjName__TINARetRetailerInitial__i_RetailerInitial, ref_i_RetailerInitial_usr1) | | | NOTE 1. |
| 2 | | **PCO1**__TINARetRetailerInitial__**i_RetailerInitial** **!** **pCALL**__TINARetRetailerInitial__i_RetailerInitial__**requestNamedAccess** | pCALL__i_RetailerInitial__requestNamedAccess__invs1 | | NOTE 2. |
| 3 | | START Timer1 | | | |
| 4 | | **PCO1**__TINARetRetailerInitial__**i_RetailerInitial** **?** **pRAISE**__TINARetRetailerInitial__i_RetailerInitial__**requestNamedAccess** CANCEL Timer1 | pRAISE__i_RetailerInitial__requestNamedAccess__invr2 | P | NOTE 3. |
| 5 | | ?TIMEOUT | | I | |
| **Detailed Comments** : NOTE:<br>1. Get the reference of the interface i_RetailerInitial.<br>2. Send a request on the operation requestNamedAccess of i_RetailerInitial.<br>3. An exception of e_UserPropertiesError is received. | | | | | |

CORBA environment. The later one is realized by a gateway component. The interworking between the two is supported by the Generic Compiler/Interpreter interface (GCI) [IN98]. Implementations of GCI, such as the ITEX one [TA98], produce skeletons for the actual adaptation of generated code to the test environment. We explain the gateway and the adaptation in the following subsequently.

The gateway component is called *TTCN/CORBA gateway*, or short, *TCgate*. TCgate is designed to support the "conversation" between a message-oriented TTCN test system and an object-oriented CORBA ORB. It provides a uniform operational interface to TTCN-based test systems that are specified as proposed in the previous sections. TCgate is a generic request-level bridge (see [CO98]) using dynamic interfaces DII and DSI, and is thus independent from particular SUT types. It is also easily portable, because it uses standard CORBA mechanisms. *Figure 4* shows the relationships between a test system, TCgate, the CORBA ORB and a SUT.

The architecture of TCgate is separated in three functional parts:

**`GatewayClient`** supports test components that interact with the SUT in the client role. It uses the dynamic invocation interface (DII) to send requests to the SUT and to receive replies or exceptions from the SUT. The interface repository is used to obtain type information of the SUT at run-time.

**`GatewayServer`** supports test components that emulate behavior of objects in the server role, which are required by the SUT. It uses the dynamic skeleton interface (DSI) to accept requests from the SUT and to forward replies or exceptions to the SUT. It uses also the interface repository.

**`GatewayMain`** represents the access point of the entire gateway to the above test systems. It has the control function for `GatewayClient` and `GatewayServer`. Especially, `GatewayMain` provides mechanisms to support the communication between multiple test components specified in *concurrent TTCN*. During the test execution, each logical test component, a Main Test Component (MTC) or a Parallel Test Component (PTC) (as defined in the concurrent TTCN terminology), is associated with a `GatewayMain` instance. In this way, the transparent distribution of MTC and PTCs is supported by the infrastructure of CORBA.

The adaptation of a TTCN test system to the interface of TCgate on the GCI basis contains generic coding functions that are capable to convert typed value dynamically, without knowledge of the types at compile-time. To communicate ASPs with the SUT, it invokes the methods of the `GatewayMain` interface. The adaptation can be reused by further test suites that are developed by applying the same method we present in this paper. TCgate and the generic

adaptation part build together a distributed test environment for CORBA-based object systems.

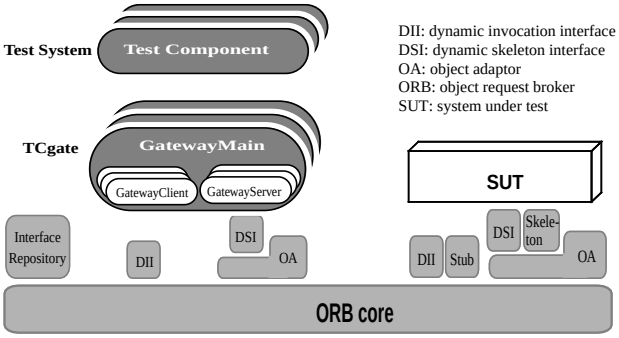


**Figure 4** The TTCN/CORBA gateway TCgate

## 7. Test Execution

The testing approach presented in this paper is demonstrated by real tests. The following systems are involved in the tests (see also Figure 4):

**SUT:** an independent implementation of the retailer domain Ret-RP on top of the *Visibroker* ORB.

**Test System:** a test system based on the ATS introduced in section 5. It comprises code produced by the *ITEX* C-code generator and a reusable adaptation to the TTCN/CORBA gateway.

**TCgate:** a generic request-level bridge that supports the integration of the non-CORBA ETS in the CORBA environment where the SUT resides. It uses the *ORBacus* ORB.

**Interface repository:** an *ORBacus* implementation of the CORBA interface repository. It provides access to type information for TCgate and the ETS at run-time.

**MSC editor (graphical):** a *Telelogic* tool for the visualization of test execution[5]. Selected test events are displayed in an MSC editor, in parallel with the test execution.

With *Figure 5* and *Figure 6*, we present the result of two tests. The MSC diagrams illustrate the temporal ordering of operation calls between the instances as well as the status of involved timers. The instance in the middle represents TCgate. The instances on the left side of `TCgate` represent the test operator and the tester. The SUT interfaces `i_RetailerInitial` and `i_RetailerNamedAccess` are indicated by `Initial` and `Access`.

The first test named `EstablishAS_1` corresponds to the scenario of *Figure 1*. The traces in *Figure 5* show a successful test: the tester sends a request through TCgate to the `Initial` interface of the SUT and starts a timer; when the excepted reply is received, the timer is stopped and the test postamble is launched; after the test postamble successfully terminates the access session, the final verdict `PASS` is given. This interaction provided by the SUT related to the `requestNamedAccess` operation is conform to the Ret-RP specification.

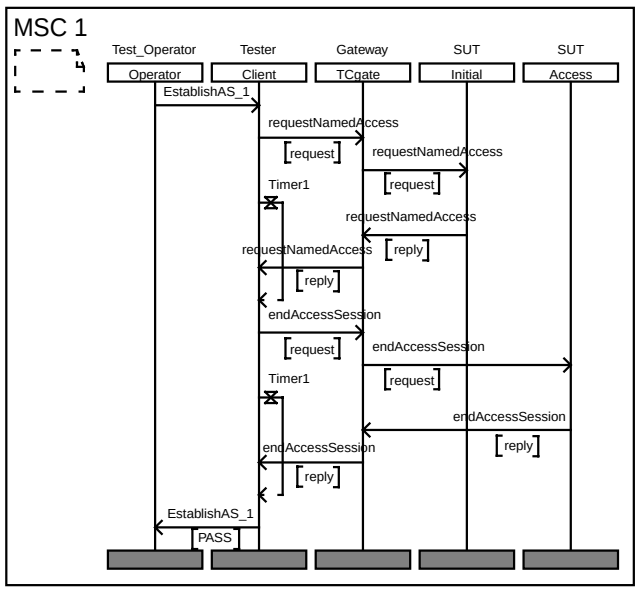


**Figure 5** MSC trace of a successful test

The second test `EstablishAS_inv2` is specified as shown in *Table 2*. Traces presented in *Figure 6* can be explained as follows: the tester sends an invalid user identification within the request to the `Access` interface of the SUT; an exception arrives before the timeout of `Timer1`; however, an `e_AuthenticationError` exception is received instead of the excepted `e_UserPropertiesError` exception (see [RT98]); this behavior is captured by the `OTHERWISE` pseudo test event, which leads to the verdict `FAIL`. The test shows a non-conform behavior of the SUT.

## 8. Summary and Future Work

This work is a contribution to the research in the area of testing *open object-based distributed systems*. It provides a practical approach to the conformance testing of TINA services.

In comparison to other earlier prototypes of distributed test systems (e.g. [HE91]), the work is based on recent concepts of RM-ODP, CORBA, TINA and TTCN. It covers both the abstract description and the execution of tests. The abstract description of tests is supported by the

5. A Telelogic library is linked to the ETS. This library uses a vendor-specific protocol called `Postmaster`. The generalization of the usage of MSC for testing will be considered in future work.

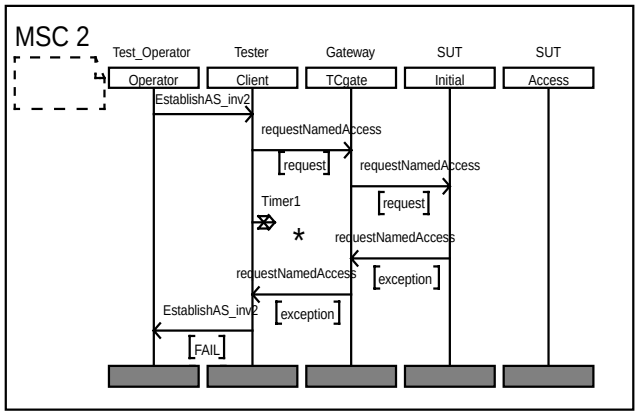


**Figure 6** MSC trace of a failed test

ODLtoTTCN mapping rules that are applied for the development of TTCN ATSs based on ODL specifications to fulfil the particular need of object-oriented systems. They build correspondence between ODL constructs and TTCN constructs, whereas standard TTCN constructs are used only. The execution of tests is supported by a gateway component TCgate, that integrates TTCN-based real test systems in the CORBA environment, which provides a distribution transparent communication with the target real systems. The usability of the approach is demonstrated by exemplary tests for the TINA Ret-RP access session.

It is shown that the ISO/ITU-T conformance testing methodology is applicable for TINA. ODLtoTTCN and TCgate are reusable, not only for TINA services, but also for other CORBA-based applications. ODLtoTTCN builds the basis for the semi-automatic test case derivation from formal specifications. They can be extended to keep track of the evolution of TINA ODL and ITU-T ODL or other computational languages. An alternative to this is the definition of new TTCN constructs, which provide directly adequate mechanisms for object systems. This alternative is left open for further study. TCgate in its current form supports TTCN including the concurrent part. It represents the entrance for TTCN-based test systems to the CORBA environment.

In a joint work, TCgate has been successfully integrated with a TSP1 (Test Synchronization Protocol 1) test manager [VT98], short *TTman*. TTman provides in particular the distribution and control of concurrent test components. TTman extends the supervisory interface of TCgate to support automated test setup and test execution for distributed systems.

In future work, we will continue the investigation of the ISO/ITU-T conformance testing methodology for distributed object systems. The next step will concentrate on testing using distributed test components, whereas the method and components obtained in this work will be applied and improved.

## 9. References


[BH97] Born, M., Hoffmann, A., Fischbeck, N., Fischer, J.: Towards a Behavioral Description of ODL, TINA Conference ‘97, Santiago de Chile, 1997.

[CO98] Object management group: The Common Object Request Broker - Architecture and Specification, revision 2.2, February 1998.

[CT97] ISO/IEC 9646: Information Technology - Open Systems Interconnection - Conformance Testing Methodology and Framework, 1997.

[HE91] Heymer, V. et al.: Konformitätssystem COAST, IIR reports 8/1991, Berlin 1991.

[IN96] INTOOL: Generic Compiler/Interpreter Interface Specification, version 2.2, GCI/NPL038, European Commission Program on Infrastructure Tools, 1996.

[KW98] Kristoffersen, F., Walter, T., TTCN: Towards a Formal Semantics and Validation of Test Suites, Computer Networks and ISDN Systems, North-Holland, 1996.

[LM98] Li, M.: Testing Computational Interfaces of TINA Services Using TTCN and CORBA, MSc Thesis, Technical University Berlin, 1998.

[LC97] Lima, L.P., Cavalli, A.R.: Test Execution of Telecommunications Services Using CORBA, FMOODS’97, Canterbury (UK) 1997.

[OD95] ITU-T X.901, ISO/IEC 10746-1: Reference Model of Open Distributed Processing 1995.

[OG97] The Open Group: Inter-domain Management - Specification Translation, preliminary specification, February 1997.

[OR98] Object-Oriented Concepts, Inc.: ORBacus Manual 1998.

[RP96] TINA Consortium: TINA Reference Point, version 3.1, June 1996.

[RT98] TINA Consortium: Ret Reference Point Specifications, version 1.0, 1998.

[SL98] Schieferdecker, I., Li, M., Hoffmann, A.: Conformance Testing of TINA Service Components - the TTCN/CORBA Gateway, IS&N’98, Antwerp (B), 1998.

[TA98] Telelogic: Tau 3.3 User Manual, 1998.

[UA97] Ulrich, A.: Test Case Generation and Test Realization in Distributed Systems, PhD Thesis (in german only), Otto-von-Güricke University Magdeburg, Germany, 1997.

[VT98] Vassiliou-Gioles, T.: Synchronization of Distributed Test Components, MSc Thesis, Technical University Berlin, 1998.