# 9

# An approach to the validation of open object-based distributed applications[1]

*S. Pickin, C. Sánchez, J. Sánchez, J.C. Yelmo,[a] J.J. Gil, E. Rodríguez[b]*
[a] *DIT, ETSI-Telecomunicación, Universidad Politécnica de Madrid. Cuidad Universitaria, 28040 Madrid, Spain. pickin@dit.upm.es*
[b] *Telefónica Investigación y Desarrollo. Emilio Vargas 6, 28043 Madrid, Spain. err@tid.es*

**Abstract**

The validation of object-based distributed applications is a difficult and as yet largely unexplored problem. We describe here a practical approach over the OMG's CORBA platform and architecture. Our approach hinges on specifying interfaces in CORBA Interface Definition Language (IDL) at other levels besides the object level and adjoining two types of complementary semantic information - then the basis of the validation methods. We then indicate how standardized FDTs such as SDL and LOTOS can be used in such a validation framework.

**Keywords**

ODP system validation, practical testing methodology, conformance testing, formal notations.

## 1 INTRODUCTION

In the ISO Open Distributed Processing (ODP) standards, it is recognized that object-based technologies constitute the most promising framework for distributed applications operating in an open heterogeneous environment. While still in the draft stage, these standards were adopted as the basis of the future telecommunications architecture by the Telecommunications Information Networking Architecture consortium (TINA-C). At the same time the Object Management Group (OMG) was defining their Common Object Request Broker Architecture (CORBA) with similar, if less far-reaching and more implementation-oriented, objectives to those of ODP. Though much of the ODP corpus has now reached the stage of international standard and numerous implementations of the CORBA standards are commercially available, validation in this setting has not yet received much attention.

[1] This work has been partly sponsored by the following Spanish research bodies: the Centre for Industrial and Technological Development (CDTI) and the Plan for Electronics and Computer Science (PEIN) as part of the OTELSO Project (EUREKA 1001) ; the Interministerial Commission for Science and Technology (CICYT) as part of the EMEDAS project.

B. Baumgarten et al. (eds.), *Testing of Communicating Systems*

This article discusses, in the context of our methodology, issues of particular relevance to object-based distributed system validation. We take a pragmatic view and draw on work of both the software and protocol validation communities. In order to carry out work with industrial relevance and in keeping with our objective of employing standards (even if *de facto*) where possible, we have worked with CORBA (and therefore deal only with operational interfaces and do not consider group communication). Our objective is to define and implement a validation environment for CORBA applications, using the Orbix[1] CORBA implementation.

## 2 VALIDATION ASPECTS OF THE DEVELOPMENT METHODOLOGY

Here we present the aspects of our methodology relevant to validation.

Our development methodology structures the application into components, with interfaces described in CORBA IDL, at different levels. Associated to the components are two types of *invocation scenarios*: external and internal. The former concern communications between the component and its peers and the second concern communications between its subcomponents. The latter will therefore include projections and decompositions of the former. Three categories of scenarios, describing obligatory, optional and forbidden behaviour, can be used. The scenarios are not intended to furnish a complete description of behaviour. Thus, a conformant implementation may legally execute traces which are not described by any of the scenarios. These may include traces with a prefix in common with a trace so described. This has obvious consequences for the discriminating power of invocation-trace testing based on these scenarios. Projections of external scenarios onto individual interfaces of a component (*interface scenarios*) are of particular importance, as explained below. The language chosen for describing the invocation scenarios is the standardized Message Sequence Chart (MSC) notation.

In this methodology, *assertions*, relating input to output, are associated to some of the external scenarios. The input (resp. output) can refer to both input (resp. output) parameters of the scenario invocations as well as to accessible aspects of the state of the IUT prior to (resp. following) the execution of the trace described in the scenario. Commonly, assertions are placed on scenarios describing single RPC calls, i.e. on CORBA two-way operations.

A set of interface scenarios and associated assertions can be bundled together with the signature of the interface (in CORBA IDL) and the joint signature of all partners appearing in the interface scenarios (also in CORBA IDL) into a development *contract*[2] [Pickin et al, 1996]. This extension of the usual notion of interface can be particularly useful for large-scale components, in particular for organizing a division of work during the design phase, for facilitating prototyping activities and for validating the service offered through a component interface.

Note that the scenarios and assertions specified in earlier phases may need to be evolved to accommodate design changes, while taking into account that modification of contractual features should involve renegotiation. Moreover, the addition of more scenarios or assertions is permitted in all phases, and in particular the testing phase, with the proviso that the contractual ones are clearly distinguished as such.

The difficulties of using current FDTs to specify object-based distributed systems, together with the extensions and modifications which are needed in order to make such specifications possible, have been treated in [Stefani, 1991], and are also discussed in [Pickin et al, 1996]. In

---

[1] Orbix is a trademark of IONA Technologies Inc.

[2] note that this notion of contract is not identical to that of ODP, TINA or other work.

view of these difficulties, our methodology only provides for FDT-based development of critical components of object-based distributed applications. This is done through the use of a CORBA-style of specification and the notion of an "FDT-wrapper" [Pickin et al, 1996].

# 3 VALIDATION OPTIONS FOR CORBA APPLICATIONS

## 3.1 Automatic (active) testing

**Preambles and postambles**: In the case of a non-FDT based component, pre-/post- amble specification is necessarily an informal procedure which may or may not use other scenarios.

In the case where the IUT is a distributed component, ensuring that it is placed in a particular state in the pre-/post- amble is a major difficulty. In such a case, we have to content ourselves with an abstract, high-level notion of state.

The distinction between the initialization of the test environment and the test preamble may not always be obvious. Generally speaking, the preparation of the environment of the IUT (setting environment permissions/variables, server registrations, dummy server activations, etc.) is part of the initialization procedure whereas the preparation of the IUT itself constitutes the preamble. However, consider IUT permissions or the case discussed in Section 4, where the IUT is to activate the test controller itself.

In the framework of the methodology discussed earlier, two types of active test can be derived:

- **Invocation-based tests**: From the invocation scenarios, tests checking that the CORBA invocations were executed in the correct order can be generated by using the MSCs expressing the obligatory and optional invocation scenarios as test purposes. These tests are evidently less discriminating than tests derived from complete specifications of behaviour; in the general case there can be no *fail* verdict, only *pass* and *inconclusive*. Nevertheless, in cases where the size and the branching of the behaviour graph are not excessive, exploration of the principal execution paths denoted by the scenarios can be of great value. Interpretation of an inconclusive verdict requires an idea of the completeness of the set of scenarios, knowledge of the problem domain and, in many cases, knowledge of the implementation.
  In the case where the IUT is a critical component whose implementation results from an FDT-based development, more discriminating tests can be derived. In this case, each message of the scenario corresponds, for example to a LOTOS action or SDL input/output. Relatively efficient tools are available for finding occurrences of the scenario in the behaviour graph of the specification[1]; after finding all the suitable ones, one reachable via the shortest path is selected. The location of this occurrence in the behaviour graph then enables a suitable preamble and postamble to be derived. A verdict of fail can then be assigned if, during the execution of the test body, a transition not foreseen by the formal specification occurs as in, for example, [Grabowski et al, 1993]. The details depend on the type of FDT wrapper.
- **Assertion-based tests** From the assertions associated to a scenario, black-box tests checking that they hold on execution of that scenario can be derived. Evidently, for true black box component testing, only those aspects of the state of the component which are verifiable using externally visible operations of that same component (where this includes reading

[1] though not always with the standardized causal semantics of the MSCs!

attribute values) should be used in the assertions. For true black box service testing, any other operations used to verify an assertion on a given operation must be visible in the same interface as that operation.
If the same scenario exists at different structural levels, the assertions associated to it at these different levels are not necessarily identical.
We derive assertion-based tests using a procedure which adapts the public domain tools of the Assertion Definition Language (ADL) project [Sankar et al, 1994] to use with CORBA.

## 3.2 Automatic passive testing or observation

We concentrate here on passive testing which takes advantage of the specifications used in our methodology. Many other dynamic properties, see the framework of [Babaoglu et al, 1995], could only be checked by ensuring causal delivery to the tester. In general, this requires the introduction of extra fields in system messages. As there are currently no OMG-standardized mechanisms for this (see [Maffeis, 1995]), such introduction would compromise the openness of the system. Two types of passive test can be derived in our methodology:

- **Invocation-based tests**: In run-time checking, the behaviour of the IUT is not originated and rigorously controlled by the tester. As the testing is based on an incomplete behaviour specification, the lack of success of a non-failure test only indicates a possible problem. To avoid being too intrusive, such testing should not signal possible problems too often. The "success" of a failure test, on the other hand, always indicates a problem. Therefore, failure tests are better suited to use in run-time checking of invocation orderings.
  The principal difficulty to be treated in invocation-based passive testing is the interleaving of the traces described in the scenarios, a phenomenon which increases with the scale of the component under test and with the number of messages in the test scenario. Though the complexity engendered by concurrency increases for larger scale components, the use of threads leads to similar problems even for the smallest scale components. Abstract interpretation of MSCs (see Section 4) reduces but does not obviate the problem. The interleaving of scenarios with a non-empty message alphabet intersection, in particular different instances of the same scenario, could still lead to failures going unnoticed by the failure tests or to a possible problem being signalled too often by other tests.
- **Assertion-based tests**: The difficulties engendered by the interleaving of the execution traces described in the scenarios mean that the assertions which are the most suited to run-time checking are those which are associated to the simplest scenarios describing single two-way operations. For these, the underlying system takes responsibility for pairing up the request and the reply. Each time such an operation is used then, its functionality is checked using the assertions. Including such tests as a permanent part of the interface facilitates implementation changes.

## 3.3 Monitoring and visualization

Here, though any verdict is completely manual, information considered to be of particular importance for determining correct behaviour can be presented so as to reflect this importance. In the framework of our methodology, two types of visualization have been defined; in both cases, user-defined multiple visualizers (implemented as CORBA servers) can show the same or different parts of the executing application (and can be executing on any host):

- **dynamic visualization of execution traces** (as MSCs): This diagram shows the invocations which have been performed between the different monitored entities, in the simplest case, CORBA servers. The sending of a local event counter value with each monitored event[1] is sufficient to ensure that the MSCs displayed respect causality (except for identical messages between the same entities which will not be a problem).
- **dynamic visualization of the application architecture** (as DARTS[2]): This diagram shows the bindings established between the stubs and interfaces of the monitored entities (currently CORBA servers). This diagram complements the execution trace visualization and can also be used as the vehicle for modifications to the monitoring session.

### 3.4 Monitoring, visualization and interactive testing

Though relatively informal, this type of debugging or operational validation is highly valued by developers. To the monitoring and visualization facilities, we can add the following:

- A facility for automatically checking a previously executed trace against a set of scenarios. Here, the boundaries of the test body are defined interactively on the MSC representation of the trace. The user is then informed whether the MSC of the executed trace corresponds fully, partially or not at all to one or more of a selected set of scenarios.
- A facility for automatically checking the current dynamic architecture against a model. We provide the capability to dynamically check that entities have been activated/instantiated and/or that bindings have been established in accordance with a previously defined model.

## 4 SOME ISSUES IN THE VALIDATION OF CORBA APPLICATIONS

**Use of MSCs**: The scenarios are described using the MSC notation, whose interpretation is as a causality diagram. However, use in object-based distributed system validation requires some extensions to the current MSC standard, in particular, the event-oriented syntax and a notational convention for denoting RPC-like calls. The first already is, and the second almost certainly will be, part of the MSC96 standard. In addition, the HMSCs of MSC96 could be used to express relations between scenarios associated to a given interface/component.

In using MSCs to specify execution traces (rather than to reflect execution traces as in monitoring), it must be decided whether they are to be interpreted as abstractions of trace segments or as complete trace segments, i.e. whether or not a conformant trace may contain other messages imbetween those specified, provided these extra messages are not members of that MSC's message alphabet[3]. We advocate use of the concrete interpretation for active testing, the abstract interpretation for passive testing and user-choice for interactive testing.

**Test architecture considerations**: Apart from the visualizers used in monitoring, the validation software is composed of a controller and a series of probes and/or dummies. The former are used in the case where we use the actual implementation of a service needed by the IUT from its environment, while the latter are used in the case where we substitute such an implementation by one with the minimal functionality needed to test the IUT. That is, whenever the

[1] along with: the identity of the entity in which the event occurs, the message identity, the event type (send or receive) and for a receive/send event, the source/destination identity respectively.

[2] DARTS is a visual specification language which is presented in [Sánchez et al, 1995].

[3] analogous to completing an FSM with loop transitions to the same state or with transitions to a fail state.

IUT behaves as a CORBA client in active or passive testing, either the real IUT environment (so probes) or the test software (so dummies) must offer the corresponding interfaces. The probes are implemented using the Orbix filters and send event notifications to the controller and/or visualizers. As mentioned previously, our purposes require that the probes insert local event counter values in the event notification messages but do not require addition of timestamps in system messages.

Location transparency in object-based distributed systems means that the controller need never itself be distributed and that the dummies can also be located at the same host, thus alleviating the sychronization problems of distributed system testing. To avoid inter-process communication within the test software, it is preferable to incorporate the dummies and the controller into a single CORBA server (Unix process), though this many not always be possible, e.g. when the IUT expects to activate different dummies in different CORBA servers. Note that in the case where the IUT expects to activate a dummy server which is implemented in the same server as the test controller, this will mean that the test will proceed with the IUT activating the test controller itself!

## 5 CONCLUSIONS

To date, validation of object-based distributed systems has not received much attention. This is in spite of the fact that, due to the efforts of the ISO ODP group and the OMG, such systems are beginning to gain widespread acceptance. The approach to the validation of CORBA applications presented here centres on the identification of components in the design phase, the description of their interface signatures in CORBA IDL, and the association of two types of semantic constraints to these components and their interfaces. This information is then used to drive the validation of the resulting implementation. In the case where FDTs are used to specify critical CORBA components, in accordance with the approach we have developed using a specification style and FDT wrappers, we point out how more satisfactory test suites for these components can be generated semi-automatically from the MSCs viewed as test purposes.

## 6 BIBLIOGRAPHY

Babaoglu, Ö., Fromentin, E., Raynal, M. (1995). *A Unified Framework for the Specification and Run-time Detection of Dynamic Properties in Distributed Computations.* Department of Computer Science, University of Bologna, Technical Report UBLCS-95-3. Jan. 1995, revised Feb 1995.

Grabowski, J., Hogrefe, D. and Nahm, R. (1993). Test case generation with test purpose specification by MSCs, in *SDL'93: Using Objects* (ed. O. Færgemand and A. Sarma), proc. SDL'93. North Holland.

Maffeis, S. (1995). Adding group communication and fault tolerance to CORBA, in Proc. of *USENIX Conference on Object-Oriented Technologies*, Monterey, CA, USA. Jun. 1995.

Pickin, S., Sánchez, C., Yelmo, J.-C., Gil, J., Rodriguez, E. (1996). Introducing formal notations in the development of object-based distributed applications, in Proc. of *First IFIP International Workshop on Formal Methods for Open Object-based Distributed Systems* (ed. Stefani, J.-B., Najm, E.), Chapman & Hall 1996.

Sánchez, J., León G., and Sánchez, C. (1995). Visual Analysis of Specifications, in Proc. of *21st EUROMICRO Conference* (ed. Kavanaugh, M.). IEEE Computer Society Press, 1995

Sankar S. and Hayes, R. (1994). ADL— an interface definition language for specifying and testing software. *ACM SIGPLAN notices*, 29(8):13–21, Jan. 1994.

Stefani, J.-B. (1991). ODP: the next target for the application of FDTs, in *Formal Description Techniques III* (ed. J. Quemada, J. Mañas and E. Vázquez), proc. FORTE'90. North Holland.

**Simon Pickin** received a Master's degree in Computer Science from Imperial College, London in 1988 having previously studied Mathematics and Theoretical Physics at Sussex University (B.Sc.), Cambridge University (Maths Tripos, Part III) and King's College, London (M.Sc.).

He spent over five years at the CNET, France Télécom's research centre, working on formal specification, protocol and telecom service engineering and particularly on verification and validation. Since 1995 he has worked as a collaborating researcher at the Departamento de Ingeniería de Sistemas Telemáticos (DIT) of the Universidad Politécnica de Madrid (UPM).

His research interests include verification and validation, formal methods and object-based distributed systems

**Carlos Antonio Sánchez Tarnawiecki** received the Electronics Engineering degree from the Universidad Nacional de Ingeniería (UNI) in Lima, Peru in 1980 and obtained his Masters degree in Communication Networks and Systems from the ETSIT of the UPM in 1990.

From 1981 to 1990 he worked as researcher and lecturer in the UNI Electrical and Electronic Engineering faculty. Since 1991 he has worked as a collaborating researcher in the software engineering group of the Telematic Systems Engineering Department (DIT) of the UPM.

His research interests include formal methods and their application in distributed systems, protocol engineering and telecommunications software.

**Jesús Sánchez Allende** received his PhD in Telecommunications Engineering from the Technical School of Telecommunications Engineering (ETSIT) of the UPM in 1994.

Since 1990 he has been working as a resercher in the software engineering group of the Telecommunications System Engineeriong Department (DIT) of the UPM. At the DIT he has been involved in R&D tasks in the fields of formal specification languages, visual formal languages, concurrent engineering, prototyping and has participated in several research projects. In 1994 he obtained a lectureship at the University Alfonso X El Sabio.

His research interests include software development environments, formal methods, open distributed processing and graphical fomalims for software design, development, analysis and testing.

**Juan-Carlos Yelmo García** received the Telecommunications Engineering degree from the Technical School of Telecommunications Engineering (ETSIT) of the UPM in 1990.

From 1988 to 1990 he worked at Telefónica I+D in the design and implementation of TMN communications software. In 1991 he joined the staff of the ETSIT in the DIT. He has partipated in several research projects at both national and international level and has published several papers mainly in the fields of FDT-based software development and technology transfer of software development methods.

His research interests include software development environments, software process modelling and enaction, open distributed platforms and architectures and formal description techniques.

**Emilio Rodriguez Rodriguez** received the Telecommunications Engineering degree in 1987 and the Specialization qualification in Switching Architectures and Data Communication in 1992 from the ETSIT of the UPM.

Since 1988 he has worked at Telefónica I+D in the development of TMN software, in the creation and maintenance of the company's software development methodology and in the provision of methodological support to software development projects. He has been active in the introduction of OO techniques in Telefonica I+D and has participated in several european collaborative projects.

His research interests include distributed object-based systems, rapid prototyping, groupware technology, workflow techniques, and the software development life-cycle.

**Juan José Gil Ríos** received the Telecommunications Engineering degree from the Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT) of the UPM in 1989.

Since 1989 he has worked in Telefónica I+D in software development methodologies and in the use of formal techniques in telecommunications systems: their application in software engineering, verification and validation, their relation with object-oriented techniques and derivation of software prototypes from formal specifications.

His research interests include distributed object-based systems, formal techniques, rapid prototyping and validation.