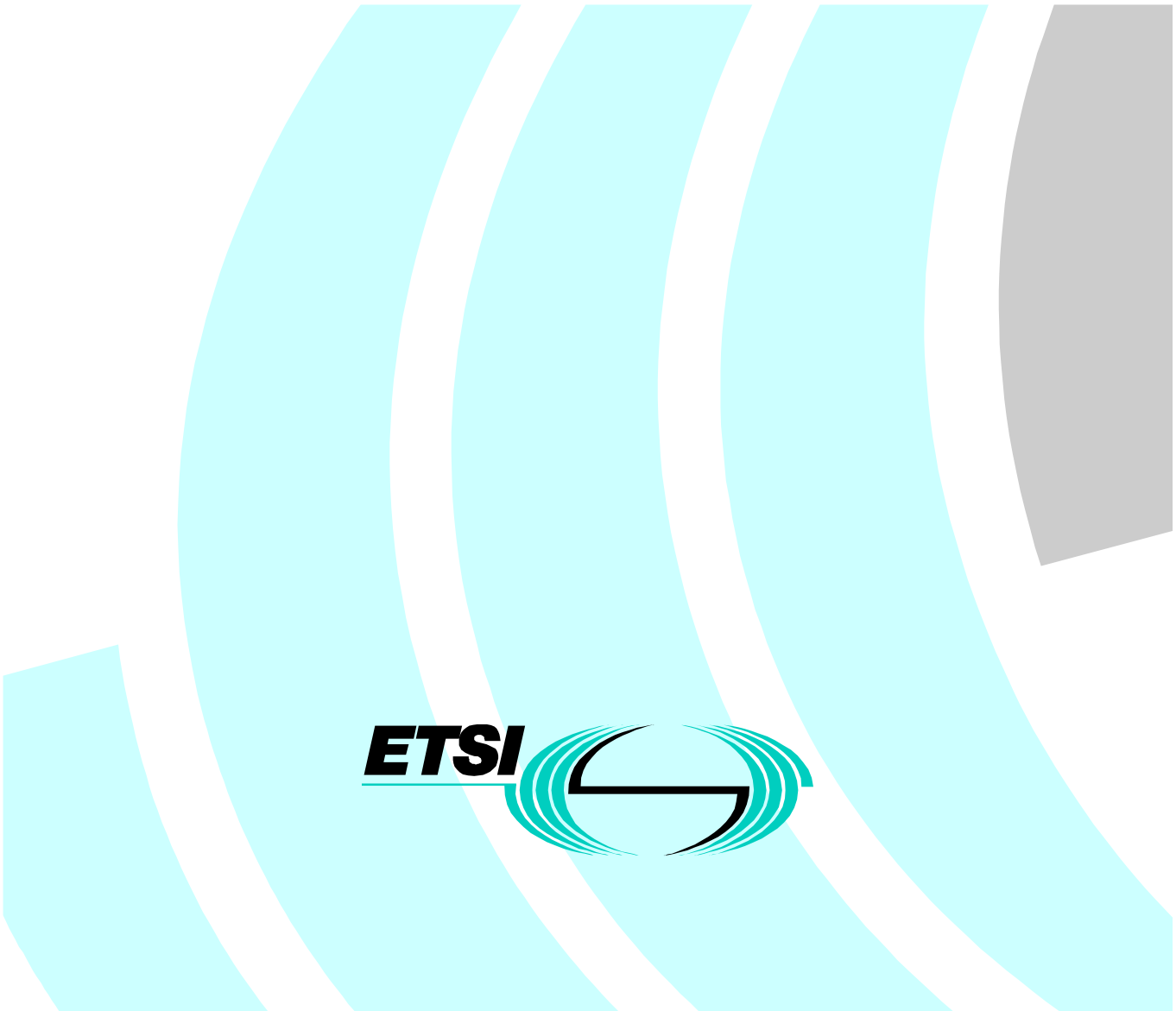
ETSI TR 102 105 V1.1.1 (1999-08)

Technical Report

**Methods for Testing and Specification (MTS);
Methodological approach to the use of object-orientation
within the standards making process;
Initial study**



Reference
DTR/MTS-00065 (jho00ics.PDF)

Keywords
Methodology, UML, SDL, MSC, TTCN

ETSI

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Internet
secretariat@etsi.fr
Individual copies of this ETSI deliverable
can be downloaded from
http://www.etsi.org
If you find errors in the present document, send your
comment to: editor@etsi.fr

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available **free of charge** from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Methods for Testing and Specification (MTS).

1 Scope

The present document provides an assessment of the feasibility of using object-orientation in the development of standards, particularly when used in association with Message Sequence Charts (MSC), Specification and Description Language (SDL) defined in ITU-T Recommendations Z.120 [11], Z.100 [9] and Z.105 [10] and the Tree and Tabular Combined Notation (TTCN) defined in ISO/IEC 9646-3 [7] for specifying the behaviour and testing of services and protocols.

A number of textual and graphical notations have been defined for object-oriented design purposes. The Guidelines for the Definition of Managed Objects (GDMO), for example, have been used extensively in the specification of international standards for telecommunication network management services. However, it is the universal nature of graphical languages which makes them particularly interesting for standardization applications. Since its introduction in 1994, the Unified Modelling Language (UML) has become one of the most popular and best defined graphical languages for object-oriented design and, for these reasons, this is the only notation considered here.

The purpose of this TR is:

- to provide a very brief introduction to the UML and the work of the Object Management Group (OMG) in standardizing it (The Universal Modelling Language [12]);
- to identify elements of the UML which could have some value if applied to the ETSI standards-making process;
- to evaluate the benefits that may be derived from their use.

The UML is considered in relation to all types of standard. On completion of this study, a set of guidelines based on a methodological approach to the use of UML in the standards-making process will be developed. These guidelines will assist technical bodies and rapporteurs to make effective use of UML wherever feasible within the process.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] ETS 300 247 (1993): "Business Telecommunications (BT); Open Network Provision (ONP) technical requirements; 2 048 kbit/s digital unstructured leased line (D2048U) Connection characteristics".

[2] ETS 300 289 (1994): "Business TeleCommunications (BTC); 64 kbit/s digital unrestricted leased line with octet integrity (D64U); Connection Characteristics".

[3] ETS 300 419 (1995): "Business TeleCommunications (BTC); 2 048 kbit/s digital structured leased lines (D2048S); Connection Characteristics".

[4] ETS 300 687 (1996): "Business TeleCommunications (BTC); 34 Mbit/s digital leased lines (D34U and D34S); Connection Characteristics".

[5] ETS 300 688 (1996): "Business TeleCommunications (BTC); 140 Mbit/s digital leased lines (D140U and D140S); Connection Characteristics".

[6] ETS 300 694 (1996): "Private Integrated Services Network (PISN); Cordless Terminal Mobility (CTM); Call handling additional network features; Service description".

[7] ISO/IEC 9646-3 (1999): "Information technology - Open Systems Interconnection - Conformance testing methodology and framework - Part 3: The Tree and Tabular Combined Notation (TTCN)".

[8] CCITT Recommendation I.130 (1988): "Method for the characterization of telecommunication services supported by an ISDN and network capabilities of an ISDN".

[9] ITU-T Recommendation Z.100 (1993): "Specification and description language (SDL)".

[10] ITU-T Recommendation Z.105 (1994): "SDL combined with ASN.1 (SDL/ASN.1)".

[11] ITU-T Recommendation Z.120 (1993): "Message Sequence Chart (MSC)".

[12] OMG Specifications ad/97-08-02 to 97-08-07 (1997): "Unified Modelling Language", Version 1.1.

[13] OMG Specification cf/96-12-03: "Meta Object Facility".

[14] OMG Specification ad/98-10-05: "XML Metadata Interchange".

[15] OMG Specification formal/98-12-01: "Common Object Request Broker Architecture", Version 2.3.

[16] OMG RFP ad/97-12-03 (1997): "Stream-based Model Interchange Format RFP".

[17] OMG RFP ad/98-11-01 (1998): "Action Semantics for the UML RFP".

[18] OMG RFP ad/99-03-13 (1999): "UML Profile for Scheduling, Performance, and Time RFP".

[19] Rumbaugh, Jacobsen & Booch: "The Unified Modelling Language Reference Manual", Addison-Wesley (1999), ISBN 0-201-30998-X.

[20] Booch, Rumbaugh & Jacobsen: "The Unified Modelling Language User Guide", Addison-Wesley (1999), ISBN 0-201-57168-4.

[21] ETS 300 012-4: "Integrated Services Digital Network (ISDN); Basic User-Network Interface (UNI); Part 4: Conformance test specification for interface IA".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following definition applies:

metamodel: a model that defines the language for expressing a model (from The UML Reference Manual [19])

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ASN.1	Abstract Syntax Notation N^{o} 1
CASE	Computer Assisted Software Engineering
CORBA	Common Object Request Broker Architecture
CTMF	Conformance Testing Methodology and Framework
GDMO	Guidelines for the Definition of Managed Objects
IS	Information System
ISDN	Integrated Services Digital Network
MAP	Member Approval Procedure
MSC	Message Sequence Chart
MV	Member Voting process
NSO	National Standards Organization
OMA	Object Management Architecture

OMG	Object Management Group
OMT	Object Modelling Technique
ONP	Open Network Provision
OO	Object-Orientation
PE	Public Enquiry process
PIXIT	Protocol Implementation eXtra Information for Testing
RFP	Requests For Proposal
ROOM	Real-Time Object-Oriented Modelling language
RTAD	Real-Time Analysis and Design
SDL	Specification and Description Language
SMIF	Stream-based Message Interchange Format
STF	Specialist Task Force
TAP	Two-step Approval Procedure
TB	ETSI Technical Body
TTCN	Tree and Tabular Combined Notation
UML	Unified Modelling Language
WG	Working Group
WI	Work Item
XMI	XML Metadata Interchange
XML	eXtensible Markup Language

4 Feasibility study

The following activities were undertaken in order to evaluate the use of object-orientation in the ETSI standardization process and these are reported here:

- review of the existing standardization process within ETSI;
- review of UML methods and standardization;
- application of UML to existing standards in three case studies:
  - a protocol standard;
  - a test specification;
  - a specification of physical characteristics;
- review of existing tool support for UML.

4.1 Standardization process

4.2 ETSI Standardization Process

Before being able to assess the value of using object-orientation in the ETSI standardization process, it is necessary to review this process itself.

The ETSI standardization process has evolved over a number of years and although methods exist and are well documented for the development of many different types of standards, those parts of the overall process which deal with the expression and maintenance of requirements are still very informal.

The process varies considerably in detail between the different Technical Bodies (TB) within ETSI although the general principles are common throughout. More established technologies generally take a more conservative approach ensuring that standards are not published until a high degree of confidence has been established in their contents. TBs responsible for newer technologies, however, often need to satisfy much tighter market constraints and are usually willing to publish standards before they have passed through such a rigorous public review phase. The whole standardization process is summarized in Table 1.

Table 1: Overview of the ETSI standardization process

No.	Step	Description
1	Proposal from one or more members for a subject of standardization	Proposals can come in various forms. Sometimes a simple proposal from one member organization but more often from a group of members who see some business potential in the particular area of standardization. It is also unusual for a single standard to be proposed. It is more likely that the proposals will be for a group of related standards (even if it is only a Stage 1, 2 and 3 for a particular service). In other cases (e.g., GSM, UMTS), the whole reason for the TB's existence is to produce a full range of standards for a specific communications technology. It is at this point that the basic requirements will first be expressed although they are likely to be at quite a high level (e.g. a secure digital air interface operating in the 1.8GHz waveband)
2	Discussion within TB/WG to determine overall/basic requirements of the standard(s)	A smaller group within a TB (usually the Sub-Technical Committee or Working Group where the proposal originated) will spend some time discussing the feasibility of the proposal and will develop further the requirements for the standard(s). These requirements are rarely documented as part of the Work Item itself. If they are recorded at all, it is more usual for them to be presented in a Technical Report as the results of a pre-normative study. Once published, the link between these detailed requirements and the resultant standards can get lost. It would be very unusual for the study report to be used as the basis for subsequent validation of the standard(s). At this stage, there will be tacit agreement on the objectives of the standardization work but a consensus commitment is not normally sought.
3	Agreement on Scope of standard	Before the Work Item can be approved and raised, the Scope of the work should be written and agreed. This should be the text that will later appear in the "Scope" clause of the resultant standard but, unfortunately, it is more often only one or two sentences giving the briefest outline of the work. The following items of information should be present in some form in the Scope: - Subject of the standard - Purpose and objective of the standard - Area of applicability of the standard - Purpose of the product or service being standardized - Limitations of the content or application of the standard - The existence of any major implementation option(s) contained in the standard - The relationship of the standard with other standards - The capabilities or limitations of upwards compatibility
4	Generation of ETSI Work Item with the support of at least 4 full members	An official Work Item needs to be raised and approved before development of the standard can proceed any further. The WI will be actively supported by at least 4 full members of ETSI (i.e., each should be willing to provide effort to write, validate or review the standard). At this point, the Scope is set in concrete and is rarely re-visited to determine whether it remains valid.
5	Allocation of WI to a rapporteur or STF	Responsibility for the actual development of the standard is allocated to a member of the committee (the rapporteur). It may not be this person that produces the draft but the rapporteur accepts responsibility for progressing the work. In most cases, the rapporteur will be one of the members that submitted the original proposal. If the skills and/or effort for the work is unavailable within the committee, it is possible that a request will be made to the ETSI Board for funding to have the standard developed by a Specialist Task Force (STF).
6	Development of schedule	The last task in raising the Work Item request is the completion of schedule details which enable the deliverable (standard) to be tracked through its various phases.
7	Start of work	The serious work of developing the standard can now begin. This will either take place at ETSI within a STF or by volunteers working at their home offices.

No.	Step	Description
8	Ongoing development of draft content	The draft content of a standard may contain text, tables, diagrams, formal specifications such as SDL, TTCN, MSC and ASN.1 or, more often, a combination of a number of these. Draft inputs are often produced by a number of contributors and the process may take many months to complete.
9	Regular reviews of drafts at WG meetings or by e-mail	An important part of the standard development process is to have the draft contents reviewed by interested and knowledgeable but reasonably independent individuals to ensure that the original requirements are being met and that the draft would be usable as a standard.
10	Formal validation if possible and desirable	Where formal notations are being used within a standard (even if they are not the normative part), tool-based validation of the specification model can be carried out. This can require the provision of time and other resources by the members of the responsible WG.
11	Consideration of draft at WG	When the rapporteur considers the draft to be essentially complete and stable, it will be reviewed for a final time by the WG to determine whether it is possible to give it their approval. The WG should be concerned mainly about the technical content of the draft.
12	Revision of draft if required	If the WG identifies any errors in the document, the rapporteur will ensure that these are rectified.
13	WG approval	When all of the required modifications have been made to the draft standard, the WG approves it and passes it to the TB for their consideration.
14	Consideration of draft at TB	The members of the TB should review the draft standard to determine whether it is possible for them to give it their approval. Although the TB should review the technical content of the draft standard, it should mainly be concerned with the "political" issues of the requirements expressed in the draft.
15	Further revision of text if required	If the TB identifies any errors in the document, the rapporteur will ensure that these are rectified.
16	TB approval	When all of the required modifications have been made to the draft standard, the TB approves it and passes it to the Secretariat to process through the necessary external approvals.
17	External approval process (MV/PE…)	Before they can have any validity, draft standards will be given external approval either by the ETSI membership (Member Approval Procedure, MAP) or by public consultation through the National Standards Organizations, NSOs, responsible for telecommunication standards in each of ETSI's member states (Two-step Approval Procedure, TAP or One-step Approval Procedure, OAP).
18	Resolution of comments	At the end of the external approval process, any comments received are assessed by the WG and/or TB. Those that are reasonable are accepted and the rapporteur is given responsibility for ensuring that the appropriate changes to the standard are made. The WG will be required to review and subsequently approve the changes if the changes are not substantial. If however, there are a significant number of changes required to the draft document, it may be necessary to repeat the approvals processes (steps 11 to 17).
19	Publication	When all approvals have been gained, the standard can be published by the Secretariat.

4.3 Introduction to Object-Orientation (OO)

4.3.1 The Unified Modelling Language (UML)

The Unified Modelling Language (UML) is a language for visualizing, specifying, constructing and documenting software systems and has been developed using best practices from existing object-oriented methods. It is standardized by Industry in the Object Management Group (OMG).

The UML uses diagrams to visualize the structures of a system in a way that transcends the textual description. The notation used in the UML has well-defined (though incomplete) syntax & semantics which are specified by OMG [12]. It can be used to specify the artefacts and decisions of analysis, design and implementation of a software-intensive architecture.

Both forward and reverse engineering are possible with the UML. Code can, at least to some degree, be generated from a UML model and a model can be reconstructed from an implementation. At a high level, it is also possible to simulate a UML system. However, more formal definition of the system's behaviour is necessary before it can be simulated fully. There are ongoing activities in OMG to extend the language in this area and these are discussed in subclause 4.3.2 of the present document.

4.3.1.1 The Architecture of a System

The architecture of a software-intensive system can have five interlocking views:

- the Use Case view:

  specifies the forces that shape the system's architecture and describes the behaviour of the system as seen by its users, analysts and testers;

- the Design view:

  collects the vocabulary (as classes, interfaces and collaborations) of the problem and its solution and shows the functional requirements of the system;

- the Process view:

  concentrates on the performance, scalability and throughput of the system and presents the threads and processes;

- the Implementation view:

  presents components that are used to assemble the system and can be used in the configuration management of the system;

- the Deployment view:

  shows the nodes that form the system’s topology and on which the system executes.

The relationship between these views is shown in Figure 1 (source: The Unified Modelling Language User Guide [20]).

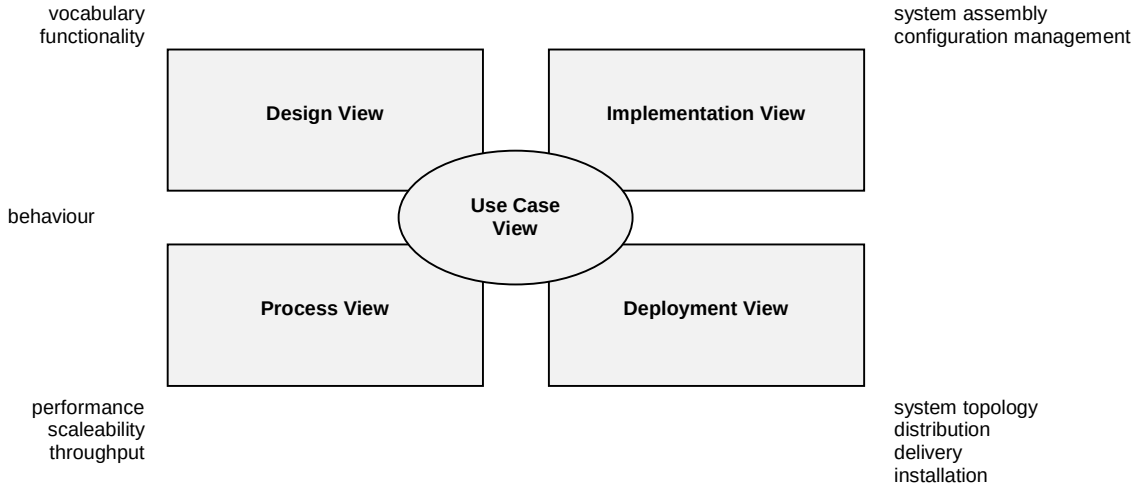


Figure 1: Relationship between views of a system

4.3.1.2 A Conceptual Model of the UML

The UML has three major elements:

- a set of basic building blocks:

  - things;
  - relationships;
  - diagrams.
- a set of rules for assembling these blocks;
- some common mechanisms (for example, to extend the UML notation).

4.3.1.2.1 Building Blocks of the UML

4.3.1.2.1.1 Things

The UML defines four different types of "thing":

- structural things:

  the nouns of a UML model. These are the static parts of a model and they represent conceptual elements of a system. Classes, interfaces, collaborations, use cases, active classes, components and nodes are structural things in the UML;

- behavioural things:

  the verbs of a UML model. These have behaviour over time and space. Interactions and state machines are basic behavioural things in the UML;

- grouping things:

  the organizational parts of the UML. Packages are currently the only grouping things in the UML;

- annotational things:

  the explanatory parts of the UML. Notes are currently the only annotational things in the UML.

4.3.1.2.1.2 Relationships

The UML supports four kinds of basic relationships as follows:

- dependency:

  a semantic relationship between two things. A change to one thing may affect the other;

- association:

  a structural relationship. An aggregation, which is a special kind of association, represents the relationship between a whole and its constituent parts;

- generalization:

  a specialization/generalization relationship in which objects of the specialized element are substitutable for objects of the generalized element. The child shares the structure and the behaviour of its parent;

- realization:

  a semantic relationship between elements. One element specifies a contract that another element carries out.

4.3.1.2.1.3 Diagrams

A diagram is a graphical presentation of a set of elements. UML has nine different diagrams which can be used to provide different views of a system's architecture. Diagrams are classified as either structural or behavioural diagrams.

4.3.1.2.1.3.1 Structural diagrams

A *class diagram* (see the example in Figure 2) shows a set of classes, interfaces and collaborations and their relationships, such as a generalization or a realization. Class diagrams illustrate the static design view of a system.

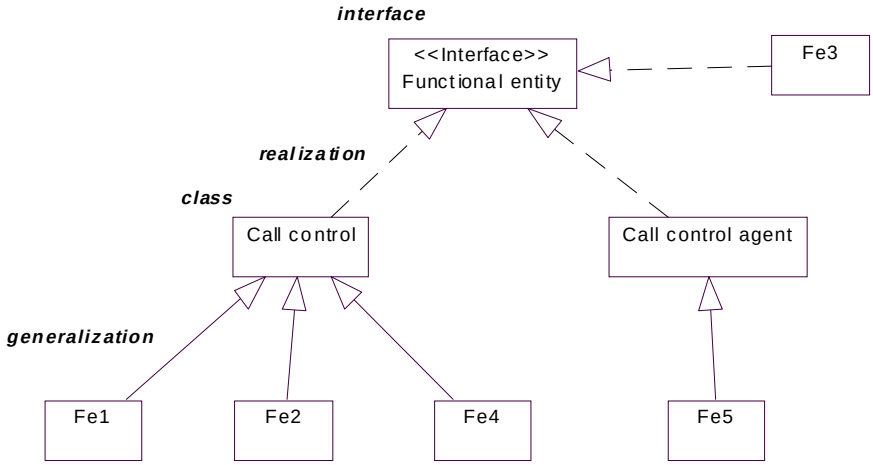


Figure 2: An example of a UML Class Diagram

An *object diagram* shows a set of objects and their relationships. Object diagrams illustrate static snapshots of instances of the classes.

A *component diagram* shows a set of components and their relationships. Component diagrams illustrate the static implementation view of a system. Figure 3 shows a simple example of a component diagram with four different components where one of these (Call control) is dependent on another (the Call control agent)

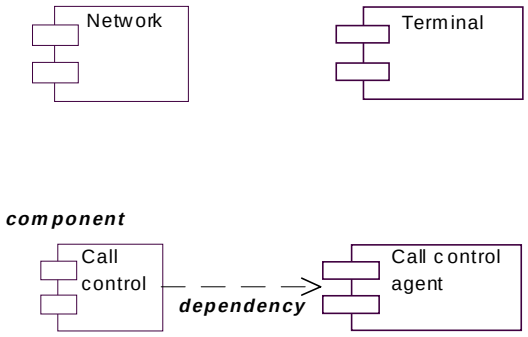


Figure 3: An example of a UML Component Diagram

A *deployment diagram* shows a set of nodes and their relationships. Deployment diagrams, as shown in Figure 4, illustrate a static deployment view of an architecture.

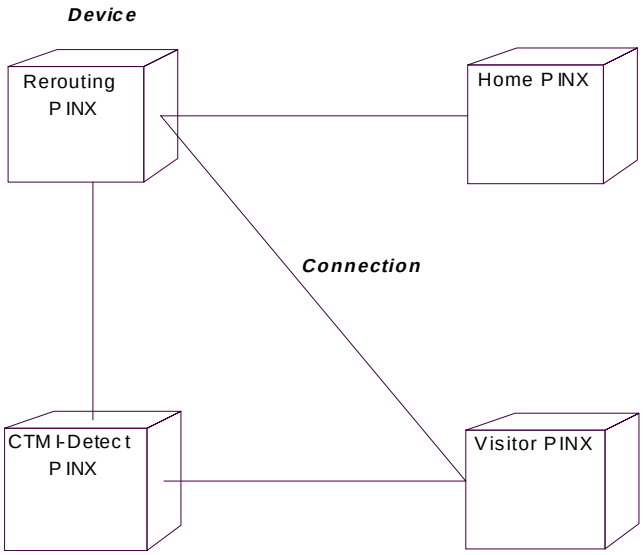


Figure 4: An example of a UML Deployment Diagram

4.3.1.2.1.3.2 Behavioural diagrams

A *use case diagram* shows a set of actors and use cases and the relationships between them. Use case diagrams, such as the one shown in Figure 5, illustrate the static use case view of a system.

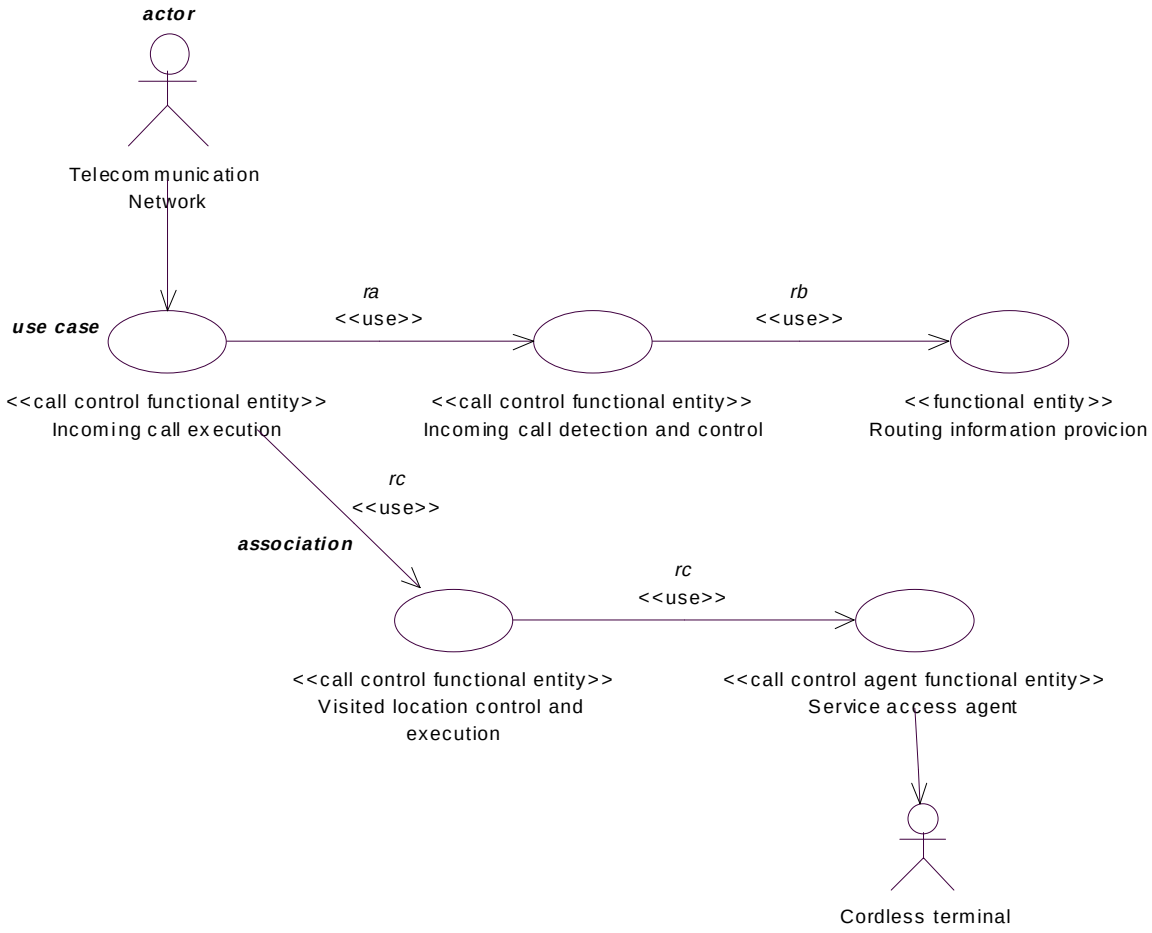


Figure 5: An example of a UML Use Case Diagram

Sequence diagrams illustrate the dynamic view of a system. As can be seen from the example in Figure 6, a sequence diagram shows a set of objects and the messages sent and received by those objects. It also emphasizes the time ordering of messages.

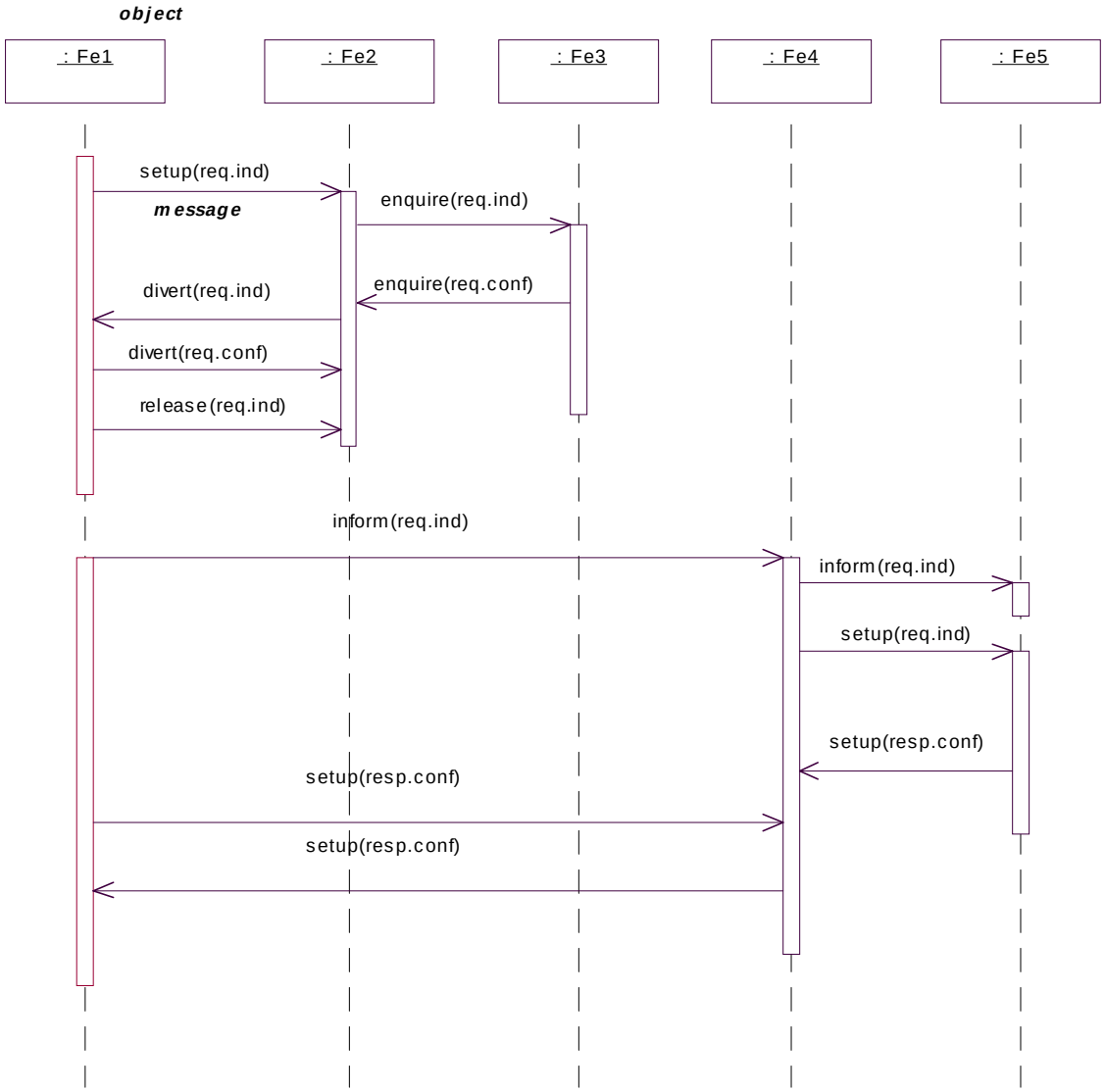


Figure 6: An example of a UML Sequence Diagram

Collaboration diagrams also illustrate the dynamic view of a system and are semantically equivalent to sequence diagrams. A collaboration diagram shows a set of objects, the links between those objects and the messages sent and received by them. It emphasizes the structural organization of the objects that send and receive messages. The collaboration diagram shown in Figure 7 is a different view of the dynamic situation shown in the sequence diagram in Figure 6.

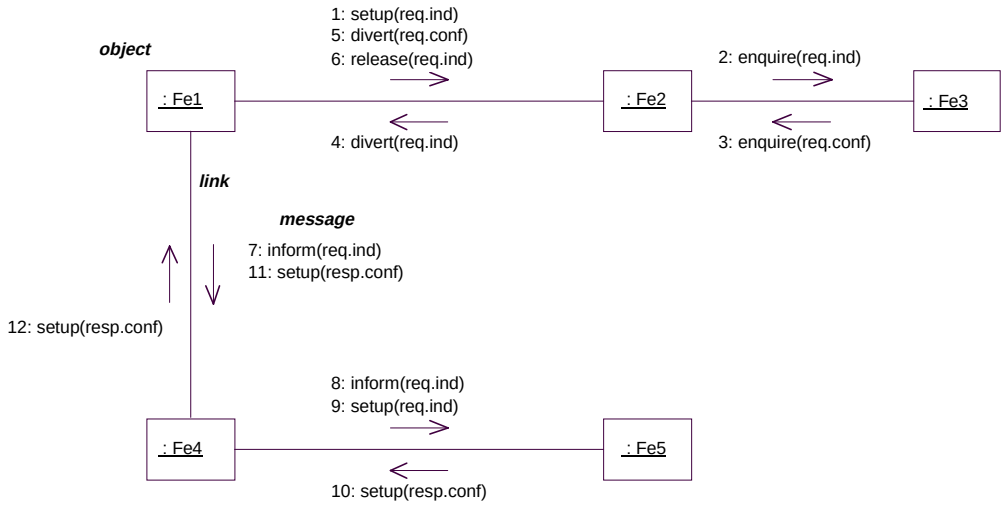


Figure 7: An example of a UML Collaboration Diagram

Statechart diagrams illustrate the dynamic view of a system. A statechart diagram shows a state machine, comprising states, transitions, events and activities. It emphasizes the event-ordered behaviour of an object. The example in Figure 8 shows how a simple process can be represented in a UML statechart.

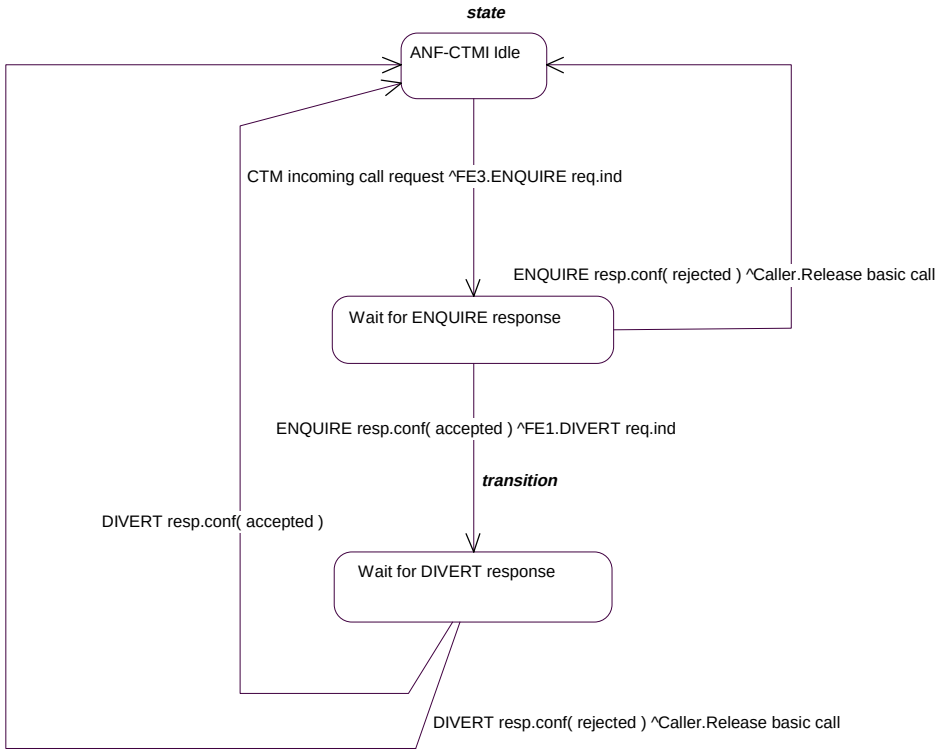


Figure 8: An example of a UML Statechart Diagram

Activity diagrams illustrate the flow of control among objects within a system. An activity diagram shows a set of activities, the sequential or branching flow from each activity to the next and objects that act and are acted upon. An example of a UML activity diagram can be seen in Figure 10.

4.3.1.2.1.3.3 Rules of UML

Within the UML there are semantic rules defined for:

- naming:

  identifies things, relationships and diagrams;

- scope:

  provides the context to give specific meaning to a name;

- visibility:

  describes how names can be seen and used by others;

- integrity:

  describes how things relate to one another properly and consistently;

- execution:

  provides the ability to run or simulate a dynamic model at a high level.

4.3.1.2.1.3.4 Common mechanisms in the UML

The UML has the following four common mechanisms that can be applied consistently:

- specifications:

  Every part of the UML graphical notation has a specification that provides a textual statement of the syntax and semantics of that building block;

- adornments:

  Every element of the UML has a basic symbol to which graphical or textual adornments specific to that symbol can be added;

- common divisions:

  In object-orientation, there are two common divisions:

  - class and object;

  - interface and implementation;

- extensibility mechanisms:

  It is possible in the UML to extend the language in controlled ways to permit the expression of a wide range of possible nuances of models across all domains. The UML's extensibility mechanisms are illustrated in Figure 9 and include:

  - *stereotypes* which extend the vocabulary of the UML, allowing the creation of new kinds of building blocks that are derived from the existing ones.

  - *tagged values* which extend the properties of a UML building block, thereby making it possible to add new information to an element's specification.

  - *constraints* which extend the semantics of a UML building block to allow the addition of new rules or the modification of existing ones.

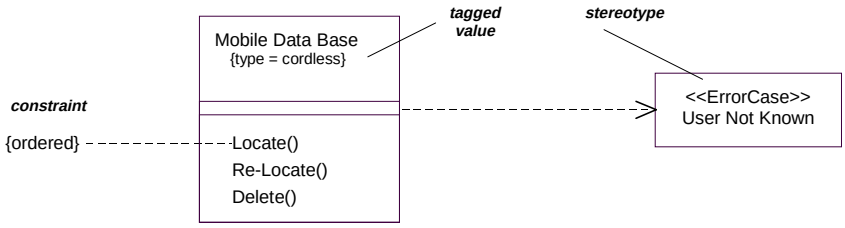


Figure 9: Examples of extensibility mechanisms

4.3.2 OMG standardization

The Object Management Group (OMG), established in 1989, is a world-wide software consortium whose goal is to introduce object-oriented standards to the software industry.

The OMG is producing standardized specifications which aim to define an object-oriented framework for distributed applications. The main achievement to date is the release of the Common Object Request Broker Architecture (CORBA) platform [15] for distributed object programming. As part of this environment, an Object Management Architecture (OMA) has been defined, providing the necessary interfaces, services and protocols to program a distributed application. As a complementary feature to the OMA, the OMG has standardized the UML [12], which is a general purpose modelling language for object analysis and design of software systems.

The continuous development of the UML is monitored by the Analysis and Design Platform Task Force within the Platform Technology Committee. The work of this task force, as that of other OMG task forces, is to clarify, amend or complement an adopted OMG specification. For this purpose, the task force generates Requests for Proposals to OMG members to submit technical proposals on a given subject. These proposals are then evaluated and some iterations take place until a proposal may be recommended for adoption by voting.

Work in progress relating to UML includes 2 main subjects addressed by the following Requests For Proposal (RFP):

RFP ad/97-12-05 [16] "Stream-based Model Interchange Format":

- a stream-based model interchange format (SMIF) to allow the file export/import of UML models. The proposal that has been retained is a format based on the XML language.

RFP ad/98-11-01 [17] "Action Semantics for UML":

- extension of the UML with an action language that will be used to specify the actions associated with state-chart transitions and the operations in class diagrams. This RFP has been issued to request proposals for the definition of semantics for the action language.

The two RFPs above should generate significant benefits for the UML user community. RFP1 will allow users to exchange their models independent of the UML tools used, provided they support the OMG standard for UML. Within the scope of making standards, this feature is very important. RFP2 will introduce missing formality to UML and thus will allow users to conduct early verification of their specification, through, for example, simulation. This feature also is very important for the validation of standards. Moreover in RFP2, it is required that the action semantics for the UML are software-platform independent in order to allow a specification to be implemented with different software technologies. The latter fact should increase reusability of specifications.

A few other tentative UML-related RFPs are being investigated by the Real-Time Analysis and Design (RTAD) sub-group, with respect, in particular, to the expression of quantitative features such as the timing constraints and resource consumption in real-time systems (RFP ad/99-03-13[18]).

4.3.3 Standardization of Object Oriented Methods within ITU-T

In working draft Recommendation Z.109 (see Bibliography), "SDL in combination with UML", ITU-T SG10 is studying the possible alignment between the UML and SDL which is defined in ITU-T Recommendation Z.100 [9]. The objective is to define a mapping of UML concepts (possibly a subset) to the SDL language (SDL-2000), so that both notations can be used in combination for system engineering. The expected benefit of this combined approach is to use the full power of UML, especially its expressiveness, for system analysis and the full power of SDL, especially its formality, for system design. It should be noted that the ITU-T Recommendation Z.109 proposal does not address the

combined use of UML and Message Sequence Charts (MSC) as defined in ITU-T Recommendation Z.120 [11]. Furthermore, ITU-T Recommendation Z.109 is based on the current definition of the UML which lacks action semantics. Therefore, in this combination, only SDL models can be used for simulation, verification and code generation.

The basis for the UML-SDL combination is to take advantage of the UML extension mechanisms. The UML-SDL mapping will be realized by using UML stereotypes, tagged values and constraints. The general approach for the mapping is to use UML packages and classes to represent the different sorts of SDL entities. However, only UML concepts that are meaningful in SDL will be supported by the combination as defined in ITU-T Recommendation Z.109. Thus, diagrams such as Collaboration Diagrams or Object Diagrams will not be supported.

The solution proposed in ITU-T Recommendation Z.109 should help users already working with older notations, such as SDL, to preserve their investments because tools (on the SDL side) can rely on it to support the migration to the combined UML-SDL solution. Although ITU-T Recommendation Z.109 may appear as a temporary solution because action semantics are being defined for the UML and may be incompatible with ITU-T Recommendation Z.109 mapping, it represents a concrete move for language experts and tool vendors to work together towards defining convergent semantics between the UML and SDL.

4.4 Case studies of three types of standard

In order to evaluate the possibility of using the UML within the ETSI standardization process, three different types of standard were reviewed as simple case studies. The types of standard considered were:

- protocol specifications;
- specifications of physical characteristics;
- conformance test specifications.

NOTE In each case, the application of the UML to a particular type of standard should be treated as an example and should not be considered in any way to be definitive as other approaches may be equally valid.

4.4.1 Protocol

4.4.1.1 Specifying protocol standards

Because the purpose of a protocol standard is to specify the behaviour of an implementation, i.e., there is some processing involved which is likely to be software-intensive, there is a clear opportunity to use object-orientation in its development. The three stage method for describing communications protocols (see CCITT Recommendation I.130 [8]) already reflects the views supported by the UML, thus:

1) Requirements capture and expression (Stage 1);
2) Functional modelling of the protocol (Stage 2);
3) Detailed description of the behaviour (Stage 3).

4.4.1.2 Requirements capture

Use cases and use case diagrams with some textual description can be used to capture the initial requirements of a protocol. As the example in Figure 5 shows, they can also help to simplify the communication of requirements between parties by defining them in a common and easily understandable language.

Stage 1 standards often use an SDL process chart to provide an overall description of the service in a graphical format. Activity diagrams can be used for the same purpose. Figure 10 shows how an activity diagram could be used to describe the incoming call service for a Cordless Terminal Mobility (CTM) user as described in ETS 300 694 [6]. The activity diagram allows the protocol or service to be described in a more general way than if constrained by the syntax and semantics of SDL which is better suited to the description of more detailed behaviour.

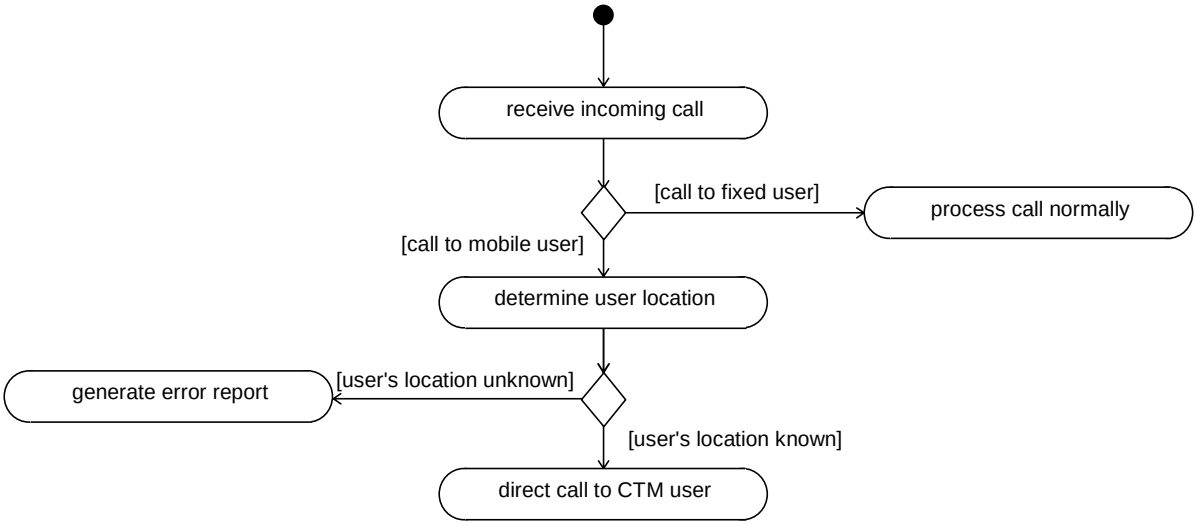


Figure 10: Example activity diagram describing CTM incoming call service

4.4.1.3 Functional modelling

As can be seen in Table 2 and in subclause 4.3, many of the diagrams used in the UML bear a strong resemblance to those already produced for most Stage 2 standards:

Table 2: Mapping of Stage 2 and UML diagrams

Stage 2 diagram	UML equivalent
functional entity model	use case & object diagram
relationship with basic service	class & collaboration diagrams
information flow diagram	sequence diagram

In addition, class diagrams can be used to specify information and function entities in the standardized system and state diagrams could be used effectively instead of the Stage 2 SDL process descriptions to describe the general behaviour of the service (see Figure 8).

4.4.1.4 Detailed description

4.4.1.4.1 Behaviour specification

Protocol behaviour can be specified using sequence, collaboration, or statechart diagrams. However, without agreed and proven action semantics, the UML cannot offer the simulation and validation capabilities of SDL and is unlikely to be able to replace it as the language of choice for the specification of normative behaviour. Current SDL tools support a transition between UML and SDL within a single specification. This will make the combined use of the two languages a genuine possibility where such an approach is likely to bring added benefits in the accuracy and understandability of a protocol standard.

4.4.1.4.2 Data specification

It is possible to specify protocol information elements using class diagrams in a similar way to that shown in Figure 11 and Figure 12 for physical characteristics. However, the UML's lack of encoding rules makes this approach less attractive than the use of ASN.1 in specifying protocol data elements which can be sent, received and understood by a range of implementations from different manufacturers.

4.4.2 Test specification

When testing the implementation of a specification, a number of tests on different layers have to be performed. On the bottom layer, conformance to physical specifications is tested; on the top layers, there are interoperability and network integration tests. ETSI has accepted some test suites for these layers as standards (e.g. ETS 300 012-4 [21]), but they are not usually developed within ETSI. For the middle layers, protocol tests are specified and this is where ETSI actively produces conformance test suites. Thus, within the remainder of this section, the consideration of UML is limited to its use and possible benefits in the specification of protocol tests.

ISO/IEC 9646-3 [7] defines a test notation, the Tree and Tabular Combined Notation (TTCN) within the Conformance Testing Methodology and Framework (CTMF).

A test suite consists of a number of test cases which are collected into test groups according to common characteristics (e.g., valid / invalid behaviour test). Test cases usually comprise a number of test steps and a test body. Unexpected behaviour is handled by at least one default test step. The essential part of every test case and test step is its behaviour description. TTCN defines basic test events for sending and receiving signals. Timers are managed with special operations to start, check and cancel timers. The corresponding timeout is also a test event.

Distributed testing is supported through the concurrent TTCN extension. A test system can consist of a number of test components. The connections between test components and from test components to the System Under Test (SUT) are described with test component configurations.

Other specifications included in a test suite are, for example, type and constraints declarations, or Protocol Implementation eXtra Information for Testing (PIXIT) statements.

4.4.2.1 Feasibility of using the UML for test specification

With the CTMF and various additional ETSI guidelines, an extensive and proven methodology exists for writing TTCN test specifications. At the moment, there is no need to change this methodology to one that incorporates the UML. The main technical reason for this is that the UML currently lacks many features needed for testing, most notably, timer support.

Nevertheless, the UML may be considered for use within the test specification process in two regards:

1) to give a visual representation of certain test suite aspects, thereby increasing the test specification readability;

2) to derive test purposes from a UML protocol specification.

4.4.2.1.1 Using the UML to increase test specification readability

ETSI test purpose specifications are usually given as textual descriptions in tables which contain the following rows:

- name;
- reference;
- purpose;
- description;
- pass criteria;
- selection;
- preamble;
- postamble.

Sometimes, informal MSCs are used to provide high level test behaviour descriptions in a graphical form. Detailed MSCs which are used for automatic test generation may be provided as an annex to the test purpose specification document.

Below is a list of the possible uses of the UML to complement the textual descriptions of test purposes. The main goal of using the UML should be to simplify the understanding of the relations between test cases and test steps (preambles and postambles) by providing a graphical view of these relations.

- Test purposes can be modelled as *class declarations*. The text which is included in the test purpose tables can be put in additional named compartments.
- Preambles and postambles can be attached to a test purpose declaration through a *dependency relationship*.
- Test grouping can be modelled using a hierarchy of *packages*.
- Test component configurations for concurrent TTCN can be specified in *deployment diagrams*.

Another possibility would be to use UML sequence diagrams instead of MSCs for the test behaviour description. At the moment, this is not feasible as current tools for automatic test generation depend on an MSC test behaviour description. Furthermore, sequence diagrams are not (yet) feature compatible with MSCs.

4.4.2.1.2 Deriving test purposes from a UML protocol specification

From a high-level, modelling point of view, test cases seem to be equivalent to UML use cases. Use cases represent the required interactions between a system and actors which are outside the system itself. It is very probable that during testing, an implementation will be checked to ensure that it meets all of the requirements. Therefore, if use cases are defined during the requirements analysis phase of the protocol specification then these use cases may be used as a starting point for test purpose specification.

In a later protocol specification stage, processes might be specified as classes with methods corresponding to signals sent to a process. The list of methods, again, may be the basis for test purpose identification.

4.4.3 Specification of physical characteristics

4.4.3.1 Standards specifying physical characteristics

Although specifications related to protocols make up the largest part of ETSI's output of standards, there are also many which specify requirements for the physical characteristics of an item claiming conformance to the standard. Such characteristics include power consumption, electromagnetic radiation, safe access to dangerous voltages and a range of transmission parameters.

4.4.3.2 Open Network Provision (ONP) leased lines

4.4.3.2.1 ONP standards

To assess the feasibility and possible benefits of using object orientation in the development of this type of standard, the range of specifications defining the connection characteristics of the digital Open Network Provision (ONP) leased lines was considered. These standards are:

- ETS 300 247 [1] 2 048 kbit/s digital unstructured leased lines (D2048U);
- ETS 300 289 [2] 64 kbit/s digital unrestricted leased line with octet integrity (D64U);
- ETS 300 419 [3] 2 048 kbit/s digital structured leased lines (D2048S);
- ETS 300 687 [4] 34 Mbit/s digital leased lines (D34U and D34S);
- ETS 300 688 [5] 140 Mbit/s digital leased lines (D140U and D140S).

There is no behaviour specified in any of these standards so there is little benefit in trying to derive sequence diagrams, collaboration diagrams and statecharts. However, since the same set of parameters need to be specified for each leased line type, it is possible to consider class and object diagrams.

4.4.3.2.2 Traditional specification of leased line characteristics

These standards already use a loose form of object-orientation by specifying the characterizing parameters of each leased line type in a table of standard entries as shown in Table 3.

Table 3: Table of empty connection attributes

Connection type attributes	**Value**	
Description	**Nature**	**Reference subclause**
Information transfer rate		
Information transfer susceptance		
Structure		
Establishment of communication		
Symmetry		
Communication configuration		
Network performance sub-attributes		
Connection type attributes	**Value**	
Description	**Nature**	**Reference subclause**
Transmission delay		
Jitter		
Octet slip		
Error parameters		
Time interval with errored blocks	**Value**	
Description	**Nature**	**Reference subclause**
Errored seconds		
Severely errored		

It can be considered that this table describes in simple terms an ONP Leased Line "class" and that the individual standards instantiate the class by giving values to the attributes to create particular leased line "objects".

Table 4 shows how the table is completed in ETS 300 289 [2] to characterize the 64 kbit/s unrestricted leased line with octet integrity (D64U).

Table 4: Completed table of connection attributes for 64 kbit/s unrestricted leased line

Connection type attributes	**Value**	
Description	**Nature**	**Reference subclause**
Information transfer rate	64 kbit/s	See 5.1.1
Information transfer susceptance	Unrestricted digital	See 5.1.2
Structure	Octet integrity	See 5.1.3
Establishment of communication	Without user intervention	See 5.1.4
Symmetry	Symmetrical in both directions	See 5.1.5
Communication configuration	Point-to-point	See 5.1.6
Network performance sub-attributes		
Connection type attributes	**Value**	
Description	**Nature**	**Reference subclause**
Transmission delay	Terrestrial and satellite options	See 5.1.7.1
Jitter	Input and output ports	See 5.1.7.2
Octet slip	5 per 24 hour period	See 5.1.7.3
Error parameters		
Time interval with errored blocks	**Value**	
Description	**Nature**	**Reference subclause**
Errored seconds	5 324 per 24 hour period	See 5.1.7.4.1
Severely errored	seconds 105 per 24 hour period	See 5.1.7.4.2

4.4.3.2.3 Use of UML to specify leased line characteristics

There are a number of ways in which this information might have been shown using UML rather than tables and Figure 11 indicates one such method. The 'Leased Line' class is shown as an aggregation of the 'Connection Type', 'Network Performance Sub-Attributes' and 'Error Parameters' classes each of which posses attributes which relate exactly to those in the three sections of the table.

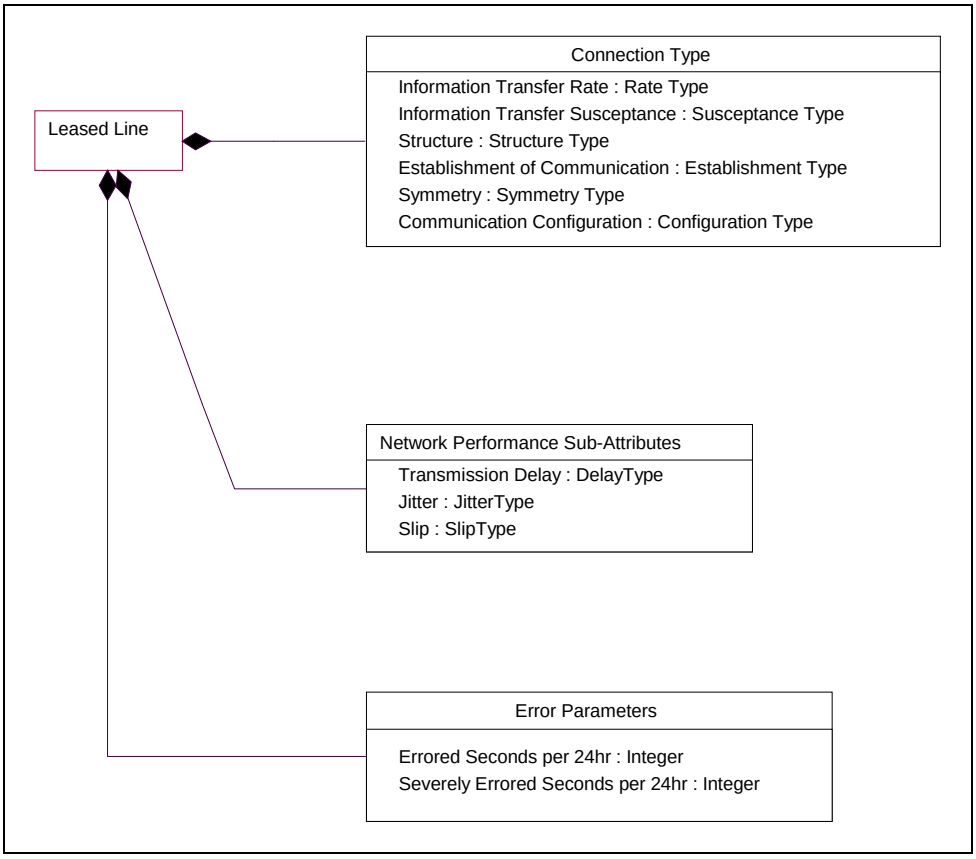


Figure 11: UML specification of an ONP Leased Line Class

On its own, this diagram does not add any additional information to that shown in Table 3 but by qualifying each of the attributes with a type, it is possible to use a further class diagram to constrain the attributes to a range of specific values. As can be seen in Figure 12, each of the types has been specified as a <<parameter type>> stereotype which is the aggregation of one of a range <<permitted value>> stereotypes. For example, any attribute of parameter type 'Structure Type' can have the value "Frame Integrity", "Octet Integrity" or "Unstructured" and no other.

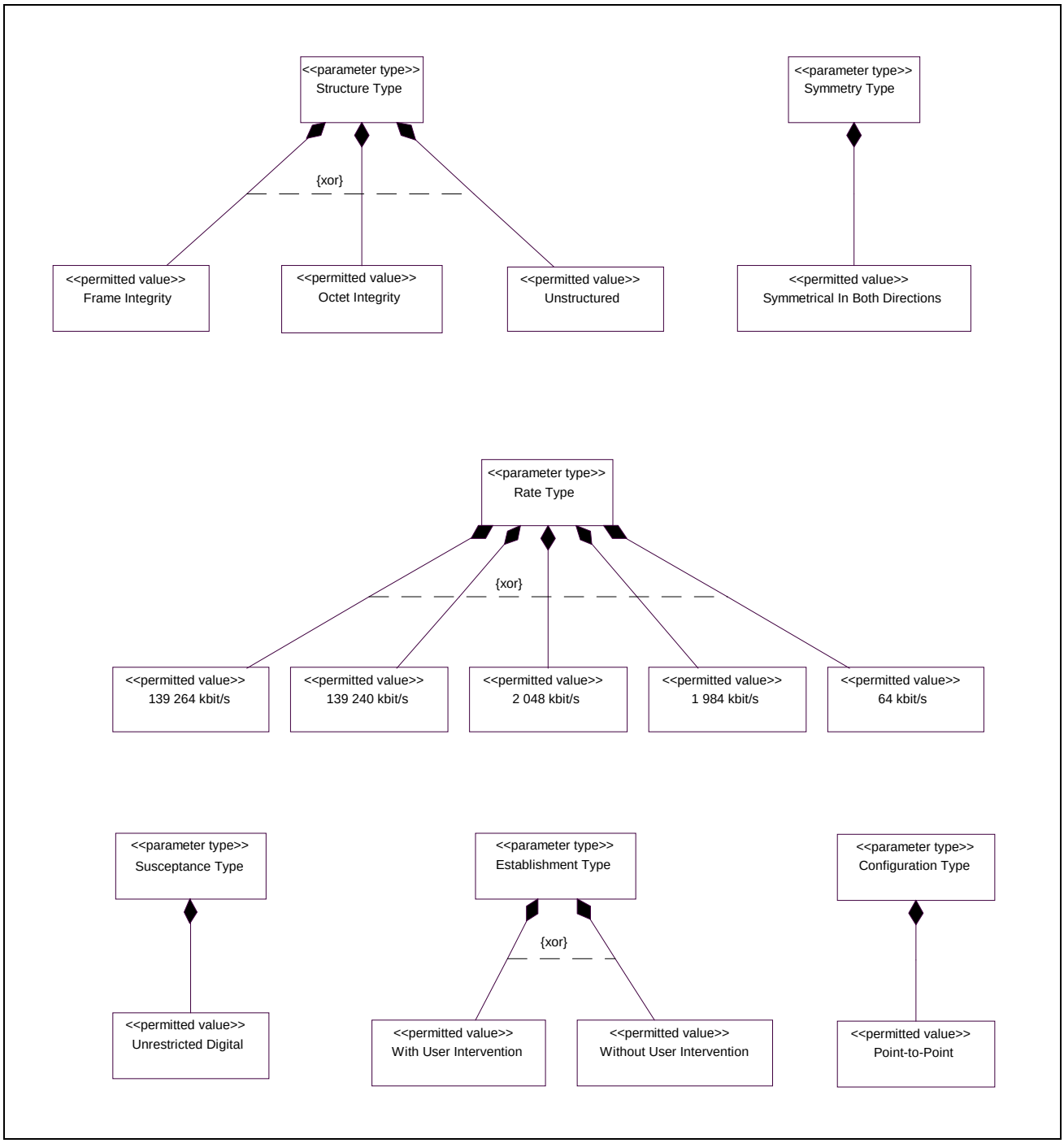


Figure 12: UML Specification of value ranges for Connection Type attributes

Having specified a general class of ONP Leased Line, it is then straightforward to create an instance of this class for each specific leased line type. Figure 13 shows an example of how a definition of the D64U leased line can be expressed as an instance of the Leased Line class.

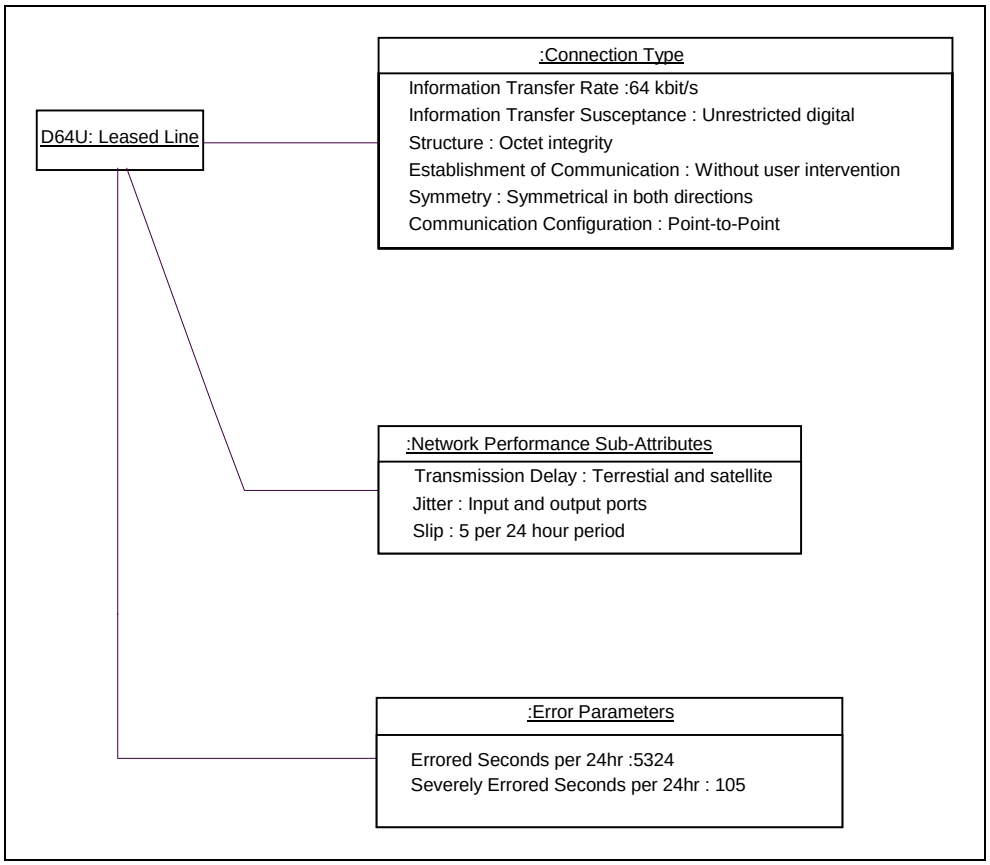


Figure 13: Object instance of D64U Leased Line

4.4.3.2.4 Summary

Without developing this model further, it is clear that the UML could be of some value in the early stages of developing standards of this type specifying physical characteristics. However, it is also arguable that the tabular presentation used in the ONP Leased Line standards is adequate for their intended use and that a more complex presentation method might require unnecessary explanation of its syntax and semantics to enable the readers to benefit fully from it.

4.5 Tool support

In the standard making process at ETSI, a number of requirements for using new methods (such as the UML) and tools should be met. They are:

- the ability to interface to existing methods & standards (MSC, SDL, TTCN, ASN.1) already in use;
- the availability of tool support and a common exchange format;
- graphical analysis and design notations.

4.5.1 Current situation

4.5.1.1 OO notations for industry

Current trends in specification techniques show that industry needs methods and tools for the following reasons:

- to automate the production of software;
- to improve its quality;
- to manage system complexity;
- to model the system at a high-level independently of any implementation.

Before UML emerged, there were several OO modelling languages having many concepts and various different interpretations. The inherent incompatibility of these languages was the main criticism from users wanting to adopt an OO modelling approach. They wanted a modelling language whose foundations are managed by both users and tool vendors in order to guarantee an agreed general purpose OO formalism.

On the tool vendor's side, the main requirement was to have a unique language semantics which fully supports an OO modelling notation. This would allow them to provide tools for model animation and verification. In addition, the impossibility of having a unique format for the exchange of OO models was seen as a restriction in the development of tool support. Obviously, there is a substantial cost induced by using and supporting various modelling environments. All these many reasons have led to a widespread consensus for UML as the convergent OO technology.

4.5.1.2 Standardization and industrialization of UML

UML unifies different modelling languages by suppressing many of the differences and redundancies. OMG's specification documents and technical papers are being elaborated continuously to provide a primary source for experts as well as for developers of supporting tools, e.g. modelling tools.

OMG support is a fundamental factor in the survival and success of UML as the industry standard object modelling language. It guarantees that, unlike the Booch or OMT languages in the past, UML will not evolve simply by following fashion or current market trends. OMG's process is rigorous, working through from RFP to Recommendation, Amendment and Adoption Vote. Although it does not prevent the making of mistakes or the influence of lobbying groups, it ensures some kind of positive inertia that is of overall benefit to the evolution of the language.

4.5.1.3 Tool support for UML

4.5.1.3.1 UML-based tools

4.5.1.3.1.1 Support for standard UML

One of the most widely-used UML tools on the market today is Rational Rose© from the Rational Software Corporation. Rose supports IS development using the UML notation for the specification and description of systems. It implements a large subset of the UML concepts including UML interfaces and diagrams, propagation of changes from one view to another, provision of capabilities for reverse engineering, and good document generation with links to requirements.

Other existing tools which offer similar capabilities to Rational Rose are as follows:

- SelectEnterprise from Select Software;
- Paradigm Plus from Platinum Technologies;
- Software through Pictures from Aonix;
- Prosa/om from Prosa Software.

4.5.1.3.1.2 Support for extended UML

The current UML notation is missing several of the concepts necessary for the high-level design of real-time systems. The Real-Time Object-Oriented Modelling language (ROOM) which is similar to the UML, offers an alternative to these limitations. However, in Rational's ObjectTime tool which supports ROOM, actions are interpreted in the statechart diagrams and missing concepts such as time constraints are added by proprietary means. These notions have yet to become part of UML and adopted by the OMG.

Other existing UML-based tools that support proprietary extensions for real-time characteristics are:

- ARTiSAN from Artisan Software;
- COOL:Jex from Sterling Software;
- Rhapsody from I-Logix.

As with any proprietary extension to a standardized language, there is always a risk that these solutions may not be compatible with the extensions that are ultimately adopted for the UML standard.

4.5.1.3.2 SDL-UML based tools

4.5.1.3.2.1 Telelogic Tau

Telelogic includes support for some UML diagrams in its Tau tool set. Specifically, class diagrams and statechart diagrams can be drawn with the Tau graphical editor. The tool set also provides means for forward engineering from UML to SDL: Classes can be translated into different kinds of type or process definitions. Class attributes are converted to SDL variable declarations, while class operations are translated into procedure references. Statecharts in turn can be converted into SDL process descriptions. Through the support of the UML, Telelogic provides an integrated development environment from requirements analysis through system specification down to target implementation.

4.5.1.3.2.2 Verilog ObjectGEODE

Verilog provides extensions, such as a class diagram editor, to its ObjectGEODE tool to support a combined UML-SDL design path. Within the tool, UML is used for high-level modelling and SDL is then used for detailed design and simulation. In a complementary approach, Verilog has developed a direct interface to the Rose tool called ObjectGEODE RoseLink. Developers are able to import into the SDL environment UML analysis models realized with Rose. Object class names, instances and their attributes are then accessible from within the ObjectGEODE internal dictionary and can be used to complete the detailed design with SDL.

4.5.1.3.3 Tool summary

To summarize, it can be said that reasonable tool support exists for the UML, although a common interchange format has yet to be implemented. The forward engineering capabilities of existing tools, such as mapping between UML and SDL, are limited. This aspect should be greatly improved when more precise semantics are defined for the UML notation. The incomplete action semantics of UML makes it difficult for an automatic tool to support animation and interpretation of models for the purpose of verification and prototyping.

4.5.2 What is needed

4.5.2.1 UML features for strong tool support

From a tool support point of view, a suitable UML notation should:

- allow users to exchange meaningful visual models

  the tools should support a common interchange format;

- provide for new integration between tools, processes and domains

  in particular the tools should support the extensibility mechanisms (e.g. tags and stereotypes) so that the user can enhance the notation;

- stay independent of any programming language or development process

  the tools should be able to generate different kinds of implementation from an UML model;

- provide a formal semantics to simplify the understanding and implementing the modelling language

  the tools should be able to provide consistent support for checking, simulation and verification;

- extend the use of OO modelling to a wide application area

tool support for the extensibility mechanisms should allow the specialization of concepts and constraints for particular application domains;

- be adopted by many users as a result of the OMG standard definition

  tool vendors should address a large and stable market;

4.5.2.2 Standardization framework for UML

As part of its mission, the OMG defines new standards for software engineering that allow the modelling of the real world through the representation of objects. Thus, initial design functionality can easily be expanded by, for example, extending some components or adding new objects to the system. This object engineering approach promoted by OMG offers faster application development and improved maintainability as well as producing reusable software.

In order to achieve these targets, the OMG is developing the necessary standard specifications, in particular:

- UML [12] to facilitate the understanding of complex models and interoperability between various CASE tools,
- MOF [13] to define a standard repository metamodel (used to represent the UML metamodel),
- XMI [14] to facilitate the exchange of information e.g. UML models between various CASE tools,
- CORBA [15] to support distributed object environments ensuring interoperability between different SW and/or HW platforms.

4.5.2.3 Action semantics for UML

The provision of an action semantics for UML is the subject of a current RFP aimed at extending UML by the definition of consistent action semantics for statechart transitions and method bodies in the objects. For the UML tool providers, this will allow them to develop verification tools such as simulators to validate the specification. Code generators and other tools are also likely to benefit from the definition of precise semantics. For the UML specifiers, the action semantics will guarantee that they are making models which are:

- interoperable;
- can be exchanged between different UML tools;
- have a behaviour that will be interpreted the same way by every simulator.

However it will be noted that, at a first stage, the action semantics are only likely to be optional for UML models so that a tool vendor may not support the action semantics but will still be compliant with UML.

Formally defining the meaning of the actions and operations used in UML models will provide for a unique interpretation of the behaviour of the model by all the supporting tools, thus facilitating their interoperability.

4.5.2.4 XMI interchange format for UML

XML Metadata Interchange (XMI) is a proposal to an OMG RFP on Stream-based Model Interchange Format (SMIF). The purpose of SMIF is to allow the interchange of models and thus the interchange of information between UML modelling tools and/or metadata repositories based on the OMG MOF standard. It should be stressed that XMI is a transfer format, i.e., any data repository or tool that can encode and decode XMI streams can exchange metadata with other repositories or tools with the same capability. XMI will allow developers of distributed systems to exchange object models and other metadata in the form of streams or files with a standard format based on XML (the eXtensible Markup Language, a W3C standard for the Internet).

4.5.2.5 Tool interoperability and model extensibility

A combination of tools from different vendors is often necessary to design a system and to document the developed models and programs. However, in practice, such combinations are difficult to achieve because of poor interchange capabilities which require translation or manual re-entry of information, with the inherent risk of loss and error. XMI tackles the problem of tool interoperability by providing a flexible and parsable information interchange format. Thus, a

tool needs only to be able to save and load the data it uses in XMI format in order to interoperate with other XMI capable tools.

The extent of information that can be exchanged between two tools is limited by how much of the information can be understood by both tools. If both share the same metamodel, all of the information transferred can be understood and used. In practice, it is likely that only a subset of the metamodel may be shared, with each tool adding its own extensions.

Moreover, having a shared model is not enough on its own. Each tool vendor needs to be able to extend the information content of the model to include pieces of information that have not been included in the shared model. XMI allows a vendor to attach additional information to shared definitions in a way that allows the information to be preserved and passed through a tool that does not understand the information. Using the extension mechanism, an XMI stream can be passed from tool to tool without any information loss.

4.5.2.6 SDL-UML alignment and tool support

In respect of ETSI's needs in the standard making process, the mission of a tool manufacturer is to provide its customers with integrated solutions and tools that enhance their productivity, reduce time-to-market and improve the quality of their software. The basic principles of the UML definition meet these needs.

Reliance on standards is a key part of such a strategy as can be seen from the tool support for SDL, MSC, TTCN or OMT. Moreover, in this context, UML is an emerging standard that provides a smooth evolution of OMT. Thus the UML technology has become a strategic direction for vendors of tools dedicated to the development of distributed intensive software systems.

To enable industrial customers to preserve their investments made with other notations such as OMT, MSC, SDL and TTCN, the tool vendors will support a smooth migration from these notations to the UML solution. It is essential that they work together in order to converge on compatible semantics as far as possible.

5 Conclusions

It was not obvious from the summary in Table 1 how and where object orientation would fit into the overall ETSI standards-making process. However, by studying the potential use of the UML in the three most significant types of standard (protocol specifications, the definition of physical characteristics and the specification of test suites) it is clear that object-orientation based on this language could be used within the ETSI standards-making process. Use of the UML should, in many cases, provide additional benefits to those achieved with the methods currently in use.

The greatest benefits of using the UML are likely to be realized at the earlier stages of standards development as the language excels in the formalization and expression of requirements but currently lacks the precision and maturity of other notations such as SDL, ASN.1 and TTCN.

Although object-oriented methods (mainly GDMO) have traditionally been used in the specification of telecommunications network management services, they have rarely been used in other areas of standardization. To make the best use of the UML across the full spectrum of ETSI's activities, there would need to be a significant change in the general methodology of standards-making such that the overall process becomes "requirements-based" rather than "results-based". In the first instance, the UML could be used as a background tool without any of its constituent diagrams ever appearing in published standards. Such use of the UML within this methodology would, in itself, yield considerable benefits.

Support of the UML in automatic tools is growing fast and, most importantly, is already available in the established SDL tools although they do not offer the full range of UML diagrams. Other tools dedicated solely to the UML are available but few of these support the complete graphical set. However, the tools are likely to grow with the language such that a lack of tool capability is unlikely to hamper the use of the UML by ETSI.

With a formal graphical language such as the UML, it is essential that a simple set of guidelines is available to standards writers in order to realize the full range of benefits that are available through the intelligent and consistent use of the language.

Annex A (informative): Useful WWW addresses

The following URLs can be accessed for further information on UML publications and tools:

The Object Management Group	http://www.omg.org
Verilog	http://www.verilogusa.com
Telelogic	http://www.telelogic.se
Rational Software Corporation	http://www.rational.com
Select Software	http://www.selectst.com
Platinum Technologies	http://www.platinum.com
Aonix Software	http://www.aonix.com
Prosa Software	http://www.prosa.fi
Artisan Software	http://www.artisansw.com
Sterling Software	http://www.sterling.com
I-Logix	http://www.ilogix.com

Bibliography

The following material, though not specifically referenced in the body of the present document (or not publicly available), gives supporting information.

- ITU-T Recommendation Z.109: "SDL in combination with UML".
- Schneider & Winters: "Applying Use Cases", Addison-Wesley (1998), ISBN 0-201-3-981-5.
- Jacobsen, Booch & Rumbaugh: "The Unified Software Development Process", Addison-Wesley (1999), ISBN 0-201-57169-2.

History

Document history		
V1.1.1	August 1999	Publication

ISBN 2-7437-3267-9
Dépôt légal : Août 1999