INTERNATIONAL STANDARD

ISO/IEC 19793

First edition
2008-12-15

Information technology — Open Distributed Processing — Use of UML for ODP system specifications

Technologies de l'information — Traitement réparti ouvert — Utilisation de l'UML pour les specifications de système ODP

Reference number
ISO/IEC 19793:2008(E)

CONTENTS

Page

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 19793 was prepared jointly by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering* in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.906.

0 Introduction

The rapid growth of distributed processing has led to the adoption of the Reference Model of Open Distributed Processing (RM-ODP), which provides a coordinating framework for the standardization of Open Distributed Processing (ODP). RM-ODP creates an architecture within which support of distribution, interworking, and portability can be integrated. This architecture provides a framework for the specification of ODP systems.

RM-ODP is based on precise concepts derived from current distributed processing developments and, as far as possible, on the use of formal description techniques for specification of the architecture. It does not recommend any notation.

The Unified Modeling Language™ (UML®) was developed by the Object Management Group™ (OMG™). It provides a notation for modelling in support of information system design and is widely used throughout the IT industry as the language and notation of choice.

This Recommendation | International Standard refines and extends the definition of how ODP systems are specified by defining the use of the Unified Modeling Language for the expression of ODP system specification.

0.1 RM-ODP

The RM-ODP consists of:

– Part 1 (ITU-T Rec. X.901 | ISO/IEC 10746-1): Overview, which contains a motivational overview of ODP, giving scoping, justification and explanation of key concepts, and an outline of the ODP architecture. It contains explanatory material on how the RM-ODP is to be interpreted and applied by its users, who may include standards writers and architects of ODP systems. It also contains a categorization of required areas of standardization expressed in terms of the reference points for conformance identified in ITU-T Rec. X.903 | ISO/IEC 10746-3. Part 1 is not normative.

– Part 2 (ITU-T Rec. X.902 | ISO/IEC 10746-2): Foundations, which contains the definition of the concepts and analytical framework for normalized description of (arbitrary) distributed processing systems. It introduces the principles of conformance to ODP standards and the way in which they are applied. This is only to a level of detail sufficient to support ITU-T Rec. X.903 | ISO/IEC 10746-3 and to establish requirements for new specification techniques. Part 2 is normative.

– Part 3 (ITU-T Rec. X.903 | ISO/IEC 10746-3): Architecture, which contains the specification of the required characteristics that qualify distributed processing as open. These are the constraints to which ODP standards shall conform. It uses the descriptive techniques from ITU-T Rec. X.902 | ISO/IEC 10746-2. Part 3 is normative.

– Part 4 (ITU-T Rec. X.904 | ISO/IEC 10746-4): Architectural semantics, which contains a formalization of the ODP modelling concepts defined in clauses 8 and 9 of ITU-T Rec. X.902 | ISO/IEC 10746-2. The formalization is achieved by interpreting each concept in terms of the constructs of one or more of the different standardized formal description techniques. Part 4 is normative.

In the same series as the RM-ODP are a number of other Standards and Recommendations, and, of these, the principal one that concerns this Recommendation | International Standard is:

– The Enterprise Language (ITU-T Rec. X.911 | ISO/IEC 15414), which refines and extends the enterprise language defined in ITU-T Rec. X.903 | ISO/IEC 10746-3 to enable full enterprise viewpoint specification of an ODP system.

0.2 UML

The Unified Modeling Language (UML) is a visual language for specifying and documenting the artifacts of systems. It is a general-purpose modelling language that can be used with all major object and component methods and that can be applied to all application domains (e.g., health, finance, telecom, aerospace) and implementation platforms (e.g., J2EE, CORBA®, .NET).

The version of UML currently adopted as an International Standard (ISO/IEC 19501) is UML 1.4, which is basically the language that was originally adopted by the OMG in the 1990s. UML was substantially extended by the OMG in 2005 to produce version 2, which offers significant enhancements, particularly in the way the language and notation handle structured classifiers. These enhancements have been found to be essential for expressing many of the more complex concepts in the RM-ODP computational and engineering language. As a result, this Recommendation | International Standard takes UML version 2 as its baseline.

UML version 2 has been structured modularly, with the ability to select only those parts of the language that are of direct interest. It is extensible, so it can be easily tailored to meet the specific user requirements. The UML specification defines thirteen types of diagram, divided in two categories that represent, respectively: the static structure of the objects in a system (structure diagrams) and the dynamic behaviour of the objects in a system (behaviour diagrams). In addition, UML incorporates extension mechanisms that allow the definition of new dialects of UML (managed using UML profiles) to customize the language for particular platforms and domains.

The UML specification is defined using a metamodelling approach (i.e., a metamodel is used to specify the model that comprises UML). That metamodel has been constructed so that the resulting family of UML languages is fully aligned with the rest of the OMG specifications (e.g., MOF™, OCL, XMI®) and to allow the exchange of models between tools.

0.3 Overview and motivation

ITU-T Rec. X.903 | ISO/IEC 10746-3 defines a framework for the specification of ODP systems comprising:

a) five viewpoints, called enterprise, information, computational, engineering and technology, which provide a basis for the specification of ODP systems;

b) a viewpoint language for each viewpoint, defining concepts and rules for specifying ODP systems from the corresponding viewpoint.

This Recommendation | International Standard defines:

– use of the viewpoints prescribed by the RM-ODP to structure UML system specifications;

– rules for expressing RM-ODP viewpoint languages and specifications with UML and UML extensions (e.g., UML profiles).

It allows UML tools to be used to process viewpoint specifications, facilitating the software design process.

Currently there is growing interest in the use of UML for system modelling. However, there is no widely agreed approach to the structuring of such specifications. This adds to the cost of adopting the use of UML for system specification, hampers communication between system developers and makes it difficult to relate or merge system specifications where there is a need to integrate IT systems.

The RM-ODP defines essential concepts necessary to specify open distributed processing systems from five prescribed viewpoints and provides a framework for the structuring of specifications for distributed systems.

However, the RM-ODP prescribes neither a notation, nor a model development method.

This Recommendation | International Standard provides the necessary framework for ODP system specification using UML. It defines both a UML based notation for the expression of such specifications, and an approach for their structuring using the notation, thus providing the basis for model development methods.

By defining how UML and UML extensions should be used to express RM-ODP viewpoint specifications, the standard enables the ODP viewpoints and ODP architecture to provide the needed framework for system specification using UML.

This Recommendation | International Standard contains the following annexes:

– Annex A: An example of ODP specifications using UML.

This annex is not normative.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Open distributed processing – Use of UML for ODP system specifications

1 Scope

This Recommendation | International Standard defines use of the Unified Modeling Language (UML 2.1.1 Superstructure Specification, OMG document formal/07-02-05) for expressing system specifications in terms of the viewpoint specifications defined by the Reference Model of Open Distributed Processing (RM-ODP, ITU-T Recs X.901 to X.904 | ISO/IEC 10746 Parts 1 to 4) and the Enterprise Language (ITU-T Rec. X.911 | ISO/IEC 15414). It covers:

a) the expression of a system specification in terms of RM-ODP viewpoint specifications using defined UML concepts and extensions (e.g., structuring rules, technology mappings, etc.);

b) relationships between the resultant RM-ODP viewpoint specifications.

This document is intended for the following audiences:

– ODP modellers who want to use the UML notation for expressing their ODP specifications in a graphical and standard way;

– UML modellers who want to use the RM-ODP concepts and mechanisms to structure their UML system specifications; and

– modelling tool suppliers, who wish to develop UML-based tools that are capable of expressing RM-ODP viewpoint specifications.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

– ITU-T Recommendation X.725 (1995) | ISO/IEC 10165-7:1996, *Information technology – Open Systems Interconnection – Structure of management information: General Relationship Model*.

– ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2:1996, *Information technology – Open Distributed Processing – Reference Model: Foundations*.

– ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3:1996, *Information technology – Open Distributed Processing – Reference Model: Architecture*.

– ITU-T Recommendation X.904 (1997) | ISO/IEC 10746-4:1998, *Information technology – Open Distributed Processing – Reference Model: Architectural semantics*.

– ITU-T Recommendation X.911 (2001) | ISO/IEC 15414:2002, *Information technology – Open Distributed Processing – Reference model – Enterprise language*.

2.2 OMG specifications

– *OMG Unified Modeling Language: Superstructure, version 2.1.1*, formal/07-02-05.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.1 Definitions from ODP standards

3.1.1 Modelling concept definitions

This Recommendation | International Standard makes use of the following terms as defined in ITU-T Rec. X.902 | ISO/IEC 10746-2:

abstraction; action; activity; architecture; atomicity; behaviour (of an object); binding; class; client object; communication; composition; component object [2-5.1]; composite object; configuration (of objects); conformance point; consumer object; contract; creation; data; decomposition; deletion; distributed processing; distribution transparency; <X> domain; entity; environment; environment contract; epoch; error; establishing behaviour; failure; fault; <X> group; identifier; information; initiating object; instance; instantiation (of an <X> template); internal action; interaction; interchange reference point; interface; interface signature; interworking reference point; introduction; invariant; location in space; location in time; name; naming context; naming domain; notification; object; obligation; ODP standards; ODP system; open distributed processing; perceptual reference point; permission; persistence; producer object; programmatic reference point; prohibition; proposition; quality of service; reference point; refinement; role; server object; spawn action; stability; state (of an object); subdomain; subtype; supertype; system; <X> template; term; terminating behaviour; trading; type (of an <X>); viewpoint (on a system).

3.1.2 Viewpoint language definitions

This Recommendation | International Standard makes use of the following terms as defined in ITU-T Rec. X.903 | ISO/IEC 10746-3:

binder; capsule; channel; cluster; community; computational behaviour; computational binding object; computational object; computational interface; computational viewpoint; dynamic schema; engineering viewpoint; distributed binding; enterprise object; enterprise viewpoint; <X> federation; information object; information viewpoint; interceptor; invariant schema; node; nucleus; operation; protocol object; static schema; stream; stub; technology viewpoint; <viewpoint> language.

3.2 Definitions from the Enterprise Language

This Recommendation | International Standard makes use of the following terms as defined in ITU-T Rec. X.911 | ISO/IEC 15414:

actor (with respect to an action); agent; artefact (with respect to an action); authorization; commitment; community object; declaration; delegation; evaluation; field of application (of a specification); interface role; objective (of an <X>); party; policy; prescription; principal; process; resource (with respect to an action); scope (of a system); step; violation.

3.3 Definitions from the Unified Modeling Language

This Recommendation | International Standard makes use of the following terms as defined in OMG document formal/07-02-05:

abstract class; action; activity; activity diagram; aggregate; aggregation; association; association class; association end; attribute; behaviour; behaviour diagram; binary association; binding; call; class; classifier; classification; class diagram; client; collaboration; collaboration occurrence; communication diagram; component; component diagram; composite; composite structure diagram; composition; concrete class; connector; constraint; container; context; delegation; dependency; deployment diagram; derived element; diagram; distribution unit; dynamic classification; element; entry action; enumeration; event; exception; execution occurrence; exit action; export; expression; extend; extension; feature; final state; fire; generalizable element; generalization; guard condition; implementation; implementation class; implementation inheritance; import; include; inheritance; initial state; instance; interaction; interaction diagram; interaction overview diagram; interface; internal transition; lifeline; link; link end; message; metaclass; metamodel; method; multiple classification; multiplicity; n-ary association; name; namespace; node; note; object; object diagram; object flow state; object lifeline; operation; package; parameter; parent; part; partition; pattern; persistent object; pin; port; postcondition; precondition; primitive type; profile; property; pseudo-state; realization; receive [a message]; receiver; reception; refinement; relationship; role; scenario; send [a message]; sender; sequence diagram; signal; signature; slot; state;

state machine diagram; state machine; static classification; stereotype; stimulus; structural feature; structure diagram; subactivity state; subclass; submachine state; substate; subpackage; subsystem; subtype; superclass; supertype; supplier; tagged value; time event; time expression; timing diagram; trace; transition; type; usage; use case; use case diagram; value; visibility.

3.4 Definitions from ODP standards refined or extended in this Recommendation | International Standard

This Recommendation | International Standard refines or extends the following terms from ITU-T Rec. X.902 | ISO/IEC 10746-2, ITU-T Rec. X.903 | ISO/IEC 10746-3, or ITU-T Rec. X.911 | ISO/IEC 15414:

– Policy (see [7.1.3]).

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply.

MOF	Meta Object Facility
OCL	Object Constraint Language
ODP	Open Distributed Processing
OMG	Object Management Group
RM-ODP	Reference Model of Open Distributed Processing
UML	Unified Modeling Language
XMI	XML Metadata Interchange

NOTE – UML, CORBA, XMI, MOF, OMG, Object Management Group, and Unified Modeling Language are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

5 Conventions

In the text that follows, the following conventions apply.

This Recommendation | International Standard is referred to as "this document".

ITU-T Rec. X.902 | ISO/IEC 10746-2 (RM-ODP Part 2: Foundations) and ITU-T Rec. X.903 | ISO/IEC 10746-3 (RM-ODP Part 3: Architecture) are referred to as "Part 2" and "Part 3" of the RM-ODP, respectively.

ITU-T Rec. X.911 | ISO/IEC 15414 (RM-ODP Enterprise Language) is referred to as "the Enterprise Language".

The UML Superstructure Specification (see [2.2]) is referred to as "the UML specification". The UML notation defined in the UML specification is referred to as "UML".

References to the normative text of this document, to the text of Parts 2 and 3 of the RM-ODP, to the Enterprise Language and to UML are expressed in one of these forms:

[n.n]	– a reference to clause n.n of this document.
[Part 2 – n.n]	– a reference to clause n.n of RM-ODP Part 2;
[Part 3 – n.n]	– a reference to clause n.n of RM-ODP Part 3;
[E/L – n.n]	– a reference to clause n.n of the Enterprise Language;
[UML – n.n]	– a reference to clause n.n of the UML specification.

For example, [Part 2 – 9.4] is a reference to subclause 9.4 of Part 2 of the RM-ODP; and [6.5] is a reference to clause 6.5 of this document. These references are for the convenience of the reader.

NOTE – The clauses correspond to the specific dated versions of the documents referenced in clause 2.

In the clauses that follow, except in the headings, terms in *italic* face are terms of the RM-ODP viewpoint languages as defined in Parts 2 and 3 of the RM-ODP, or in the Enterprise Language. UML concepts are shown in sans-serif typeface. UML stereotype names are shown in normal font, enclosed in guillemets (« and »).

The following conventions apply to the UML diagrams:

- Association end names are placed at the end of the association that is adjacent to the class playing the role. Association end names are omitted if they do not add meaning to the diagram. In this case, the implied association end name is the name of the class at that end of the association.
- Cardinalities of associations are placed adjacent to the class that has the cardinality.
- Where there are no attributes, the attribute part of the class box is suppressed.
- Black diamonds are used to represent whole/part associations, with no cardinality or role name at the whole end of the association, and no role name at the part end of the association. The meaning is that the part cannot exist without exactly one instance of the whole.
- Nouns are used in association end names, rather than verbs.
- Class names representing ODP concepts start with upper case.
- Arrowheads accompanying association names are avoided.
- Icons associated with stereotypes are used in some of the UML figures in this document. This is done to aid understanding, the icons are not normative.

6 Overview of modelling and system specification approach

6.1 Introduction

This clause provides an introduction to this document, covering:

- an overview of ODP system specification concepts;
- an overview of UML concepts;
- an explanation of the relationships between ODP models, the subjects of those models (universes of discourse), and the UML models that express the ODP models;
- an overview of the structuring principles for system specifications defined in the document;
- an explanation of the concept of correspondences (relationships) between viewpoint specifications.

6.2 Overview of ODP concepts (extracted from RM-ODP Part 1)

An overview of the ODP modelling concepts and the structuring rules for their use is given in RM-ODP Part 1 (ITU-T Rec. X.901 | ISO/IEC 10746-1: Overview) and the concepts and structuring rules are formally defined in RM-ODP Parts 2 and 3. The text that follows (i.e., the rest of [6.2]), is abstracted from the text in RM-ODP Part 1. RM-ODP Parts 2 and 3 are the authoritative standards, and should be followed in case of any conflict between those Parts and this clause.

The framework for system specification provided by the RM-ODP has four fundamental elements:

- an object modelling approach to system specification;
- the specification of a system in terms of separate but interrelated viewpoint specifications;
- the definition of a system infrastructure providing distribution transparencies for system applications;
- a framework for assessing system conformance.

6.2.1 Object modelling

Object modelling provides a formalization of well-established design practices of abstraction and encapsulation.

- Abstraction allows the description of system functionality to be separated from details of system implementation.
- Encapsulation allows the hiding of heterogeneity, the localization of failure, the implementation of security and the hiding of the mechanisms of service provision from the service user.

The object modelling concepts cover:

- basic modelling concepts: providing rigorous definitions of a minimum set of concepts (action, object, interaction and interface) that form the basis for ODP system descriptions and are applicable in all viewpoints;

– specification concepts: addressing notions such as type and class that are necessary for reasoning about specifications and the relations between specifications, providing general tools for design, and establishing requirements on specification languages;

– structuring concepts: building on the basic modelling concepts and the specification concepts to address recurrent structures in distributed systems, and covering such concerns as policy, naming, behaviour, dependability and communication.

6.2.2 Viewpoint specifications

A *viewpoint* (*on a system*) is an abstraction that yields a specification of the whole system related to a particular set of concerns. Five *viewpoints* have been chosen to be both simple and complete, covering all the domains of architectural design. These five viewpoints are:

– the *enterprise* viewpoint, which is concerned with the purpose, scope and policies governing the activities of the specified system within the organization of which it is a part;

– the *information* viewpoint, which is concerned with the kinds of information handled by the system and constraints on the use and interpretation of that information;

– the *computational* viewpoint, which is concerned with the functional decomposition of the system into a set of objects that interact at interfaces – enabling system distribution;

– the *engineering* viewpoint, which is concerned with the infrastructure required to support system distribution;

– the *technology* viewpoint, which is concerned with the choice of technology to support system distribution.

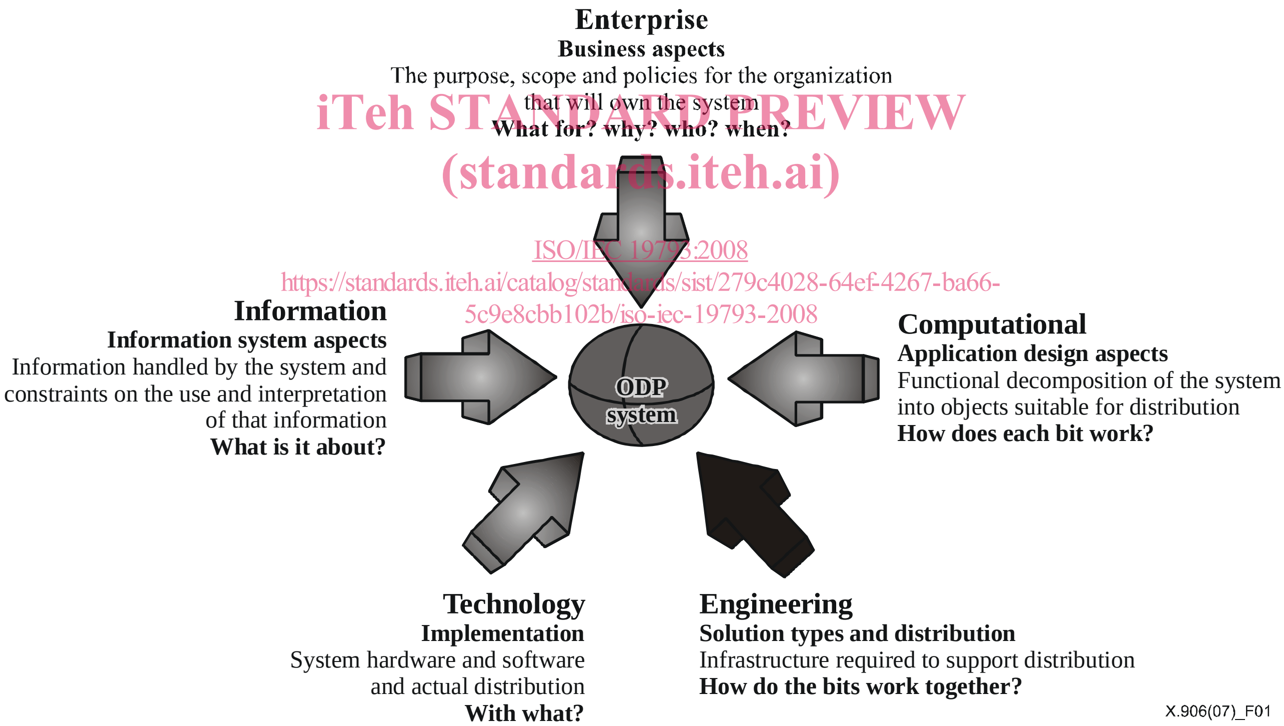

Figure 1 – RM-ODP viewpoints

For each *viewpoint* there is an associated *viewpoint language* which can be used to specify a system from that viewpoint. The object modelling concepts give a common basis for the *viewpoint languages* and make it possible to identify relationships between the different *viewpoint* specifications and to assert correspondences between the models of the system in different *viewpoints* (see [6.7]).

NOTE – Although the different viewpoints can be independently defined and there is no explicit order imposed by the RM-ODP for specifying them, a common practice is to start by developing the *enterprise* specification of the system, and then prepare the *information* and *computational* specifications. These two specifications may have constraints over each other. An iterative specification process is quite common too, whereby each *viewpoint* specification may be revised and refined as the other two are developed. Correspondences between the elements of these three *viewpoints* are defined during this process. After that, the *engineering* specification of the system is prepared, based on the *computational* specification. Correspondences between the elements of these *viewpoints* are then defined together with the newly specified elements. Finally, the *technology* specification is produced based on the *engineering* specification. Again, some refinements may be performed on the rest of the *viewpoint* specifications, due to the new requirements and constraints imposed by the particular selection of technology.

6.2.3 Distribution transparency

Distribution transparencies enable complexities associated with system distribution to be hidden from applications where they are irrelevant to their purpose. For example:

- *access transparency* masks differences of data representation and invocation mechanisms for services between systems;
- *location transparency* masks the need for an application to have information about location in order to invoke a service;
- *relocation transparency* masks the relocation of a service from applications using it;
- *replication transparency* masks the fact that multiple copies of a service may be provided in order to provide reliability and availability.

ODP standards define functions and structures to realize distribution *transparencies*. However, there are performance and cost tradeoffs associated with each *transparency* and only selected *transparencies* will be relevant in many cases. Thus, a conforming ODP system shall implement those *transparencies* that it supports in accordance with the relevant standards, but it is not required to support all *transparencies*.

6.2.4 Conformance

The basic characteristics of heterogeneity and evolution imply that different parts of a distributed system can be purchased separately, from different vendors. It is therefore very important that the behaviours of the different parts of a system are clearly defined, and that it is possible to assign responsibility for any failure to meet the system's specifications.

The framework defined to govern the assessment of conformance addresses these issues. RM-ODP Part 2 defines four classes of reference points: programmatic reference point, perceptual reference point, interworking reference point, and interchange reference point. The reference points in those classes are the candidate for conformance points. Part 2 covers:

- identification of the reference points within an architecture that provide candidate conformance points within a specification of testable components;
- identification of the conformance points within the set of viewpoint specifications at which observations of conformance can be made;
- definition of classes of conformance point;
- specification of the nature of conformance statements to be made in each viewpoint and the relation between them.

6.2.5 Enterprise language

The enterprise language provides the modelling concepts necessary to model an *ODP system* in the context of the business or organization in which it operates. An enterprise specification defines the purpose, *scope*, and *policies* of an *ODP system* and it provides the basis for checking conformance of system implementations. The purpose of the system is defined by the specified *behaviour* of the system while *policies* capture further restrictions of the *behaviour* between the system and its *environment*, or within the system itself related to the business decisions of the system owners.

NOTE 1 – An enterprise specification of a system may therefore be thought of as a statement of the "requirements" for the system. However, it must be emphasized that it is not fundamentally different from any other element of the specification for the system.

In an enterprise specification the system is modelled by one or more *enterprise objects* within the *communities* of *enterprise objects* that model its *environment*, and by the *roles* in which these objects are involved. These *roles* model, for example, the users, owners and providers of information processed by the system.

NOTE 2 – There is a question of modelling style to be considered that has particular significance for an enterprise specification, which is intended to be approachable for a subject matter expert. This is concerned with whether to name model elements in terms of instances or types. Thus it is common practice to express an enterprise specification in terms of anonymous *objects*, named by their *type*, e.g., including in enterprise specifications phrases such as "a **customer** *enterprise object* fulfils the *role* **applicant**", when what is actually meant is "an (anonymous) *enterprise object*, conforming to the *enterprise object type* **customer**, fulfils the *role* **applicant**".

6.2.6 Information language

The individual components of a distributed system should share a common understanding of the information they communicate when they interact, or the system will not behave as expected. These items of information are handled, in one way or another, by *information objects* in the system. To ensure that the interpretation of these items is consistent, the information language defines concepts for the specification of the meaning of information stored within, and manipulated by, an ODP system, independently of the way the information processing functions themselves are to be implemented.

Information held by the *ODP system* about entities in the real world, including the *ODP system* itself is modelled in an information specification in terms of *information objects*, and their relationships and *behaviour*. Basic information elements are modelled by atomic *information objects*. More complex information is modelled as composite *information objects* each modelling relationships over a set of constituent *information objects*.

The information specification comprises a set of related schemata, namely, the *invariant*, *static* and *dynamic* schemata:

– An *invariant schema* models relationships between *information objects* that must always be true, for all valid behaviours of the system.

– A *static schema* models assertions that must be true at a single point in time. A common use of static schemata is to specify the initial *state* of an *information object*.

– A *dynamic schema* specifies how the information can evolve as the system operates.

6.2.7 Computational language

The computational viewpoint is directly concerned with the distribution of processing but not with the interaction mechanisms that enable distribution to occur. The computational specification decomposes the system into *computational objects* performing individual functions and interacting at *interfaces*. It thus provides the basis for decisions on how to distribute the jobs to be done, because *objects* can be located independently assuming communications mechanisms can be defined in the engineering specification to support the behaviour at the *interfaces* to those *objects*.

The heart of the computational language is the computational object model, which constrains the computational specification by defining:

– the form of *interface* an *object* can have;

– the way that *interfaces* can be bound and the forms of *interaction* that can take place at them;

– the *actions* an *object* can perform, in particular the creation of new *objects* and *interfaces*, and the establishment of *bindings*.

The computational object model provides the basis for ensuring consistency between different engineering and technology specifications (including programming languages and communication mechanisms) since they must be consistent with the same computational object model. This consistency allows open interworking and portability of components in the resulting implementation.

The computational language enables the specifier to model constraints on the distribution of an application (in terms of *environment contracts* associated with individual *interfaces* and *interface bindings* of *computational objects*) without specifying the actual degree of distribution in the computational specification; this latter is specified in the engineering and technology specifications. This ensures that the computational specification of an application is not based on any unstated assumptions affecting the distribution of *engineering* and *technology objects*. Because of this, the configuration and degree of distribution of the hardware on which ODP applications are run can easily be altered, subject to the stated environment constraints, without having a major impact on the application software.

6.2.8 Engineering language

The engineering language focuses on the way object interaction is achieved and on the resources needed for it to take place. It defines concepts for describing the infrastructure required to support selective, distribution transparent interactions between *objects*, and rules for structuring communication *channels* between *objects*, and for structuring systems for the purposes of resource management. These rules can be modelled as *engineering templates* (for example *engineering channel template*).

Thus the computational viewpoint is concerned with when and why *objects* interact, while the engineering viewpoint is concerned with how they interact. In the engineering language, the main concern is the support of *interactions* between *computational objects*. As a consequence, there are very direct links between the viewpoint descriptions: *computational objects* are visible in the engineering viewpoint as *basic engineering objects* and computational bindings, whether implicit or explicit, are visible as either *channels* or local *bindings*.

The concepts and rules are sufficient to enable specification of internal interfaces within the infrastructure, enabling the definition of distinct conformance points for different transparencies, and the possibility of standardization of a generic infrastructure into which standardized transparency modules can be placed.

The engineering language assumes a virtual machine that corresponds to a platform offering minimal support for distribution.

NOTE – The functionality of the virtual machine assumed by the engineering language corresponds, for example, to a set of computing systems with stand-alone OS facilities plus communication facilities. In practice, the functionality available from