INTERNATIONAL STANDARD

ISO/IEC 10746-2

First edition
1996-09-15

# Information technology — Open Distributed Processing — Reference Model: Foundations

*Technologies de l'information — Traitement distribué ouvert — Modèle de référence: Fondations*

Reference number
ISO/IEC 10746-2:1996(E)

## Contents

## 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. 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.

International Standard ISO/IEC 10746-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open Systems Interconnection, data management and open distributed processing*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.902.

ISO/IEC 10746 consists of the following parts, under the general title *Information technology — Open Distributed Processing — Reference Model*:

— *Part 1: Overview*

— *Part 2: Foundations*

— *Part 3: Architecture*

— *Part 4: Architectural semantics*

## Introduction

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

The Reference Model of Open Distributed Processing (RM-ODP), ITU-T Recs. X.901 to X.904 | ISO/IEC 10746, 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.

The RM-ODP consists of:

- ITU-T Rec. X.901 | ISO/IEC 10746-1: **Overview**: 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. This part is not normative.
- ITU-T Rec. X.902 | ISO/IEC 10746-2: **Foundations**: 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. This part is normative.
- ITU-T Rec. X.903 | ISO/IEC 10746-3: **Architecture**: Contains the specification of the required characteristics that qualify distributed processing as open. These are the constraints to which ODP standards must conform. It uses the descriptive techniques from ITU-T Rec. X.902 | ISO/IEC 10746-2. This part is normative.
- ITU-T Rec. X.904 | ISO/IEC 10746-4: **Architectural semantics**: Contains a formalization of the ODP modelling concepts defined in this Recommendation | International Standard (clauses 8 and 9). The formalization is achieved by interpreting each concept in terms of the constructs of the different standardized formal description techniques. This part is normative.

This Recommendation | International Standard does not contain any annexes.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

# INFORMATION TECHNOLOGY – OPEN DISTRIBUTED PROCESSING – REFERENCE MODEL: FOUNDATIONS

## 1 Scope

This ITU-T Recommendation | International Standard covers the concepts which are needed to perform the modelling of ODP systems (see clauses 5 to 14), and the principles of conformance to ODP systems (see clause 15).

The concepts defined in clauses 5 to 15 are used in the Reference Model of Open Distributed Processing to support the definitions of:

a) the structure of the family of standards which are subject to the Reference Model;

b) the structure of distributed systems which claim compliance with the Reference Model (the configuration of the systems);

c) the concepts needed to express the combined use of the various standards supported;

d) the basic concepts to be used in the specifications of the various components which make up the open distributed system.

## 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.903 (1995) | ISO/IEC 10746-3:1996, *Information technology – Open distributed processing – Reference Model: Architecture.*

## 3 Definitions

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

### 3.1 Definitions from other Recommendations | International Standards

There are no definitions from other Recommendations | International Standards in this Recommendation | International Standard.

### 3.2 Background definitions

**3.2.1 distributed processing**: Information processing in which discrete components may be located in different places, and where communication between components may suffer delay or may fail.

**3.2.2 ODP standards**: This Reference Model and those standards that comply with it, directly or indirectly.

**3.2.3 open distributed processing**: Distributed processing designed to conform to ODP standards.

**3.2.4 ODP system**: A system (see 6.5) which conforms to the requirements of ODP standards.

**3.2.5 information**: Any kind of knowledge, that is exchangeable amongst users, about things, facts, concepts and so on, in a universe of discourse.

Although information will necessarily have a representation form to make it communicable, it is the interpretation of this representation (the meaning) that is relevant in the first place.

**3.2.6 data**: The representation forms of information dealt with by information systems and users thereof.

**3.2.7 viewpoint (on a system)**: A form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system.

## 4 Abbreviations

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

| | |
|---|---|
| ODP | Open Distributed Processing |
| OSI | Open Systems Interconnection |
| PICS | Protocol Implementation Conformance Statement |
| PIXIT | Protocol Implementation Extra Information for Testing |
| RM-ODP | Reference Model of Open Distributed Processing |
| TP | Transaction Processing |

## 5 Categorization of concepts

The modelling concepts defined in this Recommendation | International Standard are categorized as follows:

a) *Basic interpretation concepts:* Concepts for the interpretation of the modelling constructs of any ODP modelling language. These concepts are described in clause 6.

b) *Basic linguistic concepts:* Concepts related to languages; the grammar of any language for the ODP Architecture must be described in terms of these concepts. These concepts are described in clause 7.

c) *Basic modelling concepts:* Concepts for building the ODP Architecture; the modelling constructs of any language must be based on these concepts. These concepts are described in clause 8.

d) *Specification concepts:* Concepts related to the requirements of the chosen specification languages used in ODP. These concepts are not intrinsic to distribution and distributed systems, but they are requirements to be considered in these specification languages. These concepts are described in clause 9.

e) *Structuring concepts:* Concepts that emerge from considering different issues in distribution and distributed systems. They may or may not be directly supported by specification languages adequate for dealing with the problem area. Specification of objects and functions that directly support these concepts must be made possible by the use of the chosen specification languages. These concepts are described in clauses 10 to 14.

f) *Conformance concepts:* Concepts necessary to explain the notions of conformance to ODP standards and of conformance testing. These concepts are defined in clause 15.

ITU-T Recommendation X.903 | ISO/IEC 10746-3 uses the concepts in this Recommendation | International Standard to specify the characteristics for distributed processing to be open. It is organized as a set of viewpoint languages. Each viewpoint language refines concepts from the set defined in this Recommendation | International Standard. It is not necessary for all viewpoint languages to adopt the same notations. Different notations may be chosen as appropriate to reflect the requirements of the viewpoint. These notations may be natural, formal, textual or graphical. However, it will be necessary to establish correspondences between the various languages to ensure overall consistency.

## 6 Basic interpretation concepts

Although much of the ODP Architecture is concerned with defining formal constructs, the semantics of the architectural model and any modelling languages used have to be described. These concepts are primarily meta-concepts, i.e. concepts which apply generally to any form of modelling activity. It is not intended that these concepts will be formally defined, nor that they be used as the basis of formal definition of other concepts.

Any modelling activity identifies:

a) elements of the universe of discourse;

b) one or more pertinent levels of abstraction.

The elements of the universe of discourse are entities and propositions.

**6.1 Entity**: Any concrete or abstract thing of interest. While in general the word entity can be used to refer to anything, in the context of modelling it is reserved to refer to things in the universe of discourse being modelled.

**6.2 Proposition**: An observable fact or state of affairs involving one or more entities, of which it is possible to assert or deny that it holds for those entities.

**6.3 Abstraction**: The process of suppressing irrelevant detail to establish a simplified model, or the result of that process.

**6.4 Atomicity**: An entity is atomic at a given level of abstraction if it cannot be subdivided at that level of abstraction.

Fixing a given level of abstraction may involve identifying which elements are atomic.

**6.5 System**: Something of interest as a whole or as comprised of parts. Therefore a system may be referred to as an entity. A component of a system may itself be a system, in which case it may be called a subsystem.

NOTE – For modelling purposes, the concept of system is understood in its general, system-theoretic sense. The term "system" can refer to an information processing system but can also be applied more generally.

**6.6 Architecture (of a system)**: A set of rules to define the structure of a system and the interrelationships between its parts.

## 7 Basic linguistic concepts

Whatever the concepts or semantics of a modelling language for the ODP Architecture, it will be expressed in some syntax, which may include linear text or graphical conventions. It is assumed that any suitable language will have a grammar defining the valid set of symbols and well-formed linguistic constructs of the language. The following concepts provide a common framework for relating the syntax of any language used for the ODP Architecture to the interpretation concepts.

**7.1 Term**: A linguistic construct which may be used to refer to an entity.

The reference may be to any kind of entity including a model of an entity or another linguistic construct.

**7.2 Sentence**: A linguistic construct containing one or more terms and predicates; a sentence may be used to express a proposition about the entities to which the terms refer.

A predicate in a sentence may be considered to refer to a relationship between the entities referred to by the terms linked.

## 8 Basic modelling concepts

The detailed interpretation of the concepts defined in this clause will depend on the specification language concerned, but these general statements of concept are made in a language-independent way to allow the statements in different languages to be interrelated.

The basic concepts are concerned with existence and activity: the expression of what exists, where it is and what it does.

**8.1 Object**: A model of an entity. An object is characterized by its behaviour (see 8.6) and, dually, by its state (see 8.7). An object is distinct from any other object. An object is encapsulated, i.e. any change in its state can only occur as a result of an internal action or as a result of an interaction (see 8.3) with its environment (see 8.2).

An object interacts with its environment at its interaction points (see 8.11).

Depending on the viewpoint, the emphasis may be placed on behaviour or on state. When the emphasis is placed on behaviour, an object is informally said to perform functions and offer services (an object which makes a function available is said to offer a service). For modelling purposes, these functions and services are specified in terms of the behaviour of the object and of its interfaces (see 8.4). An object can perform more than one function. A function can be performed by the cooperation of several objects.

NOTES

1 The concepts of service and function are used informally to express the purpose of a piece of standardization. In the ODP family of standards, function and service are expressed formally in terms of the specification of the behaviour of objects and of the interfaces which they support. A "service" is a particular abstraction of behaviour expressing the guarantees offered by a service provider.

2 The expression "use of a function" is a shorthand for the interaction with an object which performs the function.

**8.2** **Environment (of an object)**: The part of the model which is not part of that object.

NOTE – In many specification languages, the environment can be considered to include at least one object which is able to participate without constraint in all possible interactions (see 8.3), representing the process of observation.

**8.3** **Action**: Something which happens.

Every action of interest for modelling purposes is associated with at least one object.

The set of actions associated with an object is partitioned into **internal actions** and **interactions**. An internal action always takes place without the participation of the environment of the object. An interaction takes place with the participation of the environment of the object.

NOTES

1 "Action" means "action occurrence". Depending on context, a specification may express that an action has occurred, is occurring or may occur.

2 The granularity of actions is a design choice. An action need not be instantaneous. Actions may overlap in time.

3 Interactions may be labelled in terms of cause and effect relationships between the participating objects. The concepts that support this are discussed in 13.3.

4 An object may interact with itself, in which case it is considered to play at least two roles in the interaction, and may be considered, in this context, as being a part of its own environment.

5 Involvement of the environment represents observability. Thus, interactions are observable whereas internal actions are not observable, because of object encapsulation.

**8.4** **Interface**: An abstraction of the behaviour of an object that consists of a subset of the interactions of that object together with a set of constraints on when they may occur.

Each interaction of an object belongs to a unique interface. Thus, the interfaces of an object form a partition of the interactions of that object.

NOTES

1 An interface constitutes the part of an object behaviour that is obtained by considering only the interactions of that interface and by hiding all other interactions. Hiding interactions of other interfaces will generally introduce non-determinism as far as the interface being considered is concerned.

2 The phrase "an interface between objects" is used to refer to the binding (see 13.4.2) between interfaces of the objects concerned.

**8.5** **Activity**: A single-headed directed acyclic graph of actions, where occurrence of each action in the graph is made possible by the occurrence of all immediately preceding actions (i.e. by all adjacent actions which are closer to the head).

**8.6** **Behaviour (of an object)**: A collection of actions with a set of constraints on when they may occur.

The specification language in use determines the constraints which may be expressed. Constraints may include for example sequentiality, non-determinism, concurrency or real-time constraints.

A behaviour may include internal actions.

The actions that actually take place are restricted by the environment in which the object is placed.

NOTES

1 The composition (see 9.1) of a collection of objects implicitly yields an equivalent object representing the composition. The behaviour of this object is often referred to simply as the behaviour of the collection of objects.

2 Action and activity are degenerate cases of behaviour.

3 In general, several sequences of interactions are consistent with a given behaviour.

**8.7** **State (of an object)**: At a given instant in time, the condition of an object that determines the set of all sequences of actions in which the object can take part.

Since, in general, behaviour includes many possible series of actions in which the object might take part, knowledge of state does not necessarily allow the prediction of the sequence of actions which will actually occur.

State changes are effected by actions; hence a state is partially determined by the previous actions in which the object took part.

Since an object is encapsulated, its state cannot be changed directly from the environment, but only indirectly as a result of the interactions in which the object takes part.

**8.8** **Communication**: The conveyance of information between two or more objects as a result of one or more interactions, possibly involving some intermediate objects.

NOTES

1 Communications may be labelled in terms of a cause and effect relationship between the participating objects. Concepts to support this are discussed in 13.3.

2 Every interaction is an instance of a communication.

**8.9** **Location in space**: An interval of arbitrary size in space at which an action can occur.

**8.10** **Location in time**: An interval of arbitrary size in time at which an action can occur.

NOTES

1 The extent of the interval in time or space is chosen to reflect the requirements of a particular specification task and the properties of a particular specification language. A single location in one specification may be subdivided in either time or space (or both) in another specification. In a particular specification, a location in space or time is defined relative to some suitable coordinate system.

2 By extension, the location of an object is the union of the locations of the actions in which the object may take part.

**8.11** **Interaction point**: A location at which there exists a set of interfaces.

At any given location in time, an interaction point is associated with a location in space, within the specificity allowed by the specification language in use. Several interaction points may exist at the same location. An interaction point may be mobile.

## 9 Specification concepts

### 9.1 Composition

a) (Of objects) – A combination of two or more objects yielding a new object, at a different level of abstraction. The characteristics of the new object are determined by the objects being combined and by the way they are combined. The behaviour of a composite object is the corresponding composition of the behaviour of the component objects.

b) (Of behaviours) – A combination of two or more behaviours yielding a new behaviour. The characteristics of the resulting behaviour are determined by the behaviours being combined and the way they are combined.

NOTES

1 Examples of combination techniques are sequential composition, concurrent composition, interleaving, choice, and hiding or concealment of actions. These general definitions will always be used in a particular sense, identifying a particular means of combination.

2 In some cases, the composition of behaviours may yield a degenerate behaviour, e.g. deadlock, due to the constraints on the original behaviours.

**9.2** **Composite object**: An object expressed as a composition.

### 9.3 Decomposition

a) (Of an object) – The specification of a given object as a composition.

b) (Of a behaviour) – The specification of a given behaviour as a composition.

Composition and decomposition are dual terms and dual specification activities.

**9.4** **Behavioural compatibility**: An object is behaviourally compatible with a second object with respect to a set of criteria (see Notes) if the first object can replace the second object without the environment being able to notice the difference in the objects' behaviour on the basis of the set of criteria.

Typically, the criteria impose constraints on the allowed behaviour of the environment. If the criteria are such that the environment behaves as a tester for the original object, i.e. the environment defines the smallest behaviour that does not constrain the behaviour of the original object, the resulting behavioural compatibility relation is called extension.

The criteria may allow the replacement object to be derived by modification of an otherwise incompatible object in order that it should be an acceptable replacement. An example of such a modification might be hiding of additional parameters on certain interactions. In this way, an interaction of the new object can be made to look like an interaction of the original object. In such cases behavioural compatibility is called **coerced behavioural compatibility**. If no modification is necessary, behavioural compatibility is called **natural behavioural compatibility**.

The concept of behavioural compatibility defined above on objects applies equally well to the behavioural compatibility of templates and of template types.

Behavioural compatibility is reflexive, but not necessarily symmetric or transitive (though it may be either or both).

NOTES

1 The set of criteria depends on the language in use and the testing theory applied.

2 Behavioural compatibility (with respect to a set of criteria) can be defined on template (see 9.11) and template types (see 9.19), thus:

a) if S and T are object templates, S is said to be behaviourally compatible with T if and only if any S-instantiation is behaviourally compatible with some T-instantiation (see 9.13);

b) if U and V are object template types, U and V are said to be behaviourally compatible if their corresponding templates are *behaviourally compatible*.

**9.5** **Refinement**: The process of transforming one specification into a more detailed specification. The new specification can be referred to as a refinement of the original one. Specifications and their refinements typically do not coexist in the same system description. Precisely what is meant by a more detailed specification will depend on the chosen specification language.

For each meaning of behavioural compatibility determined by some set of criteria (see 9.4), a specification technique will permit the definition of a refinement relationship. If template X refines a template Y, it will be possible to replace an object instantiated from Y by one instantiated from X in the set of environments determined by the selected definition of behavioural compatibility. Refinement relationships are not necessarily either symmetric or transitive.

**9.6** **Trace**: A record of an object's interactions, from its initial state to some other state.

A trace of an object is thus a finite sequence of interactions. The behaviour uniquely determines the set of all possible traces, but not vice versa. A trace contains no record of an object's internal actions.

**9.7** **Type (of an <X>)**: A predicate characterizing a collection of <X>s. An <X> is of the type, or satisfies the type, if the predicate holds for that <X>. A specification defines which of the terms it uses have types, i.e. are <X>s. In RM-ODP, types are needed for, at least, objects, interfaces and actions.

The notion of type classifies the entities into categories, some of which may be of interest to the specifier (see the concept of class in 9.8).

**9.8** **Class (of <X>s)**: The set of all <X>s satisfying a type (see 9.7). The elements of the set are referred to as members of the class.

NOTES

1 A class may have no members.

2 Whether the size of the set varies with time depends on the definition of the type.

**9.9** **Subtype/supertype**: A type A is a subtype of a type B, and B is a supertype of A, if every <X> which satisfies A also satisfies B.

The subtype and supertype relations are reflexive, transitive and anti-symmetric.

**9.10** **Subclass/superclass**: One class A is a subclass of another class B, and B is a superclass of A, precisely when the type associated with A is a subtype of the type associated with B.

NOTE – A subclass is by definition a subset of any of its superclasses.

**9.11** **<X> Template**: The specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it. <X> can be anything that has a type (see 9.7).

An <X> template is an abstraction of a collection of <X>s.

A template may specify parameters to be bound at instantiation time.

The definition given here is generic; the precise form of a template will depend on the specification technique used. The parameter types (where applicable) will also depend on the specification technique used.

Templates may be combined according to some calculus. The precise form of template combination will depend on the specification language used.

**9.12** **Interface signature**: The set of action templates associated with the interactions of an interface.

An object may have many interfaces with the same signature.

**9.13** **Instantiation (of an <X> template)**: An <X> produced from a given <X> template and other necessary information. This <X> exhibits the features specified in the <X> template. <X> can be anything that has a type (see 9.7).

The definition given here is generic: how to instantiate an <X> template depends on the specification language used. Instantiating an <X> template may involve actualization of parameters, which may in turn involve instantiating other <X> templates or binding of existing interfaces (see 12.4).

NOTES

1 Instantiating an action template just results in an action occurring. The phrase "instantiation of an action template" is deprecated. "Occurrence of an action" is preferred.

2 If <X> is an object, it is instantiated in its initial state. An object can participate in interactions immediately after its instantiation.

3 Instantiations from different templates may satisfy the same type. Instantiations from the same template may satisfy different types.

**9.14** **Role**: Identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object.

Specification of a template as a composition of roles enables the instantiation process to be explained as the association of a specific component of the resultant composite object with each role. The association of a component object with a role may result from the actualization of a parameter.

**9.15** **Creation (of an <X>)**: Instantiating an <X>, when it is achieved by an action of objects in the model. <X> can be anything that can be instantiated, in particular objects and interfaces.

If <X> is an interface, it is either created as part of the creation of a given object, or as an additional interface to the creating object. As a result, any given interface must be part of an object.

**9.16** **Introduction (of an <X>)**: Instantiating an <X> when it is not achieved by an action of objects in the model.

NOTES

1 An <X> can be instantiated either by creation or introduction but not both.

2 Introduction does not apply to interfaces and actions since these are always supported by objects.

**9.17** **Deletion (of an <X>)**: The action of destroying an instantiated <X>. <X> can be anything that can be instantiated, in particular objects and interfaces.

If <X> is an interface, it can only be deleted by the object to which it is associated.

NOTE – Deletion of an action is not meaningful: an action just happens.

**9.18** **Instance (of a type)**: An <X> that satisfies the type.

**9.19** **Template type (of an <X>)**: A predicate defined in a template that holds for all the instantiations of the template and that expresses the requirements the instantiations of the template are intended to fulfill.

The object template subtype/supertype relation does not necessarily coincide with behavioural compatibility. Instances of a template type need not be behaviourally compatible with instantiations of the associated template. They do coincide if:

a) a transitive behavioural compatibility relation is considered; and

b) template subtypes are behaviourally compatible with their template supertypes.

NOTES

1 This concept captures the notion of substitubility by design.

2 The form of the predicate that expresses the template type depends on the specification language used.

3 As a shorthand, "instances of a template T" are defined to be "instances of the template type associated with template T".

4 Figure 1 illustrates the relationships between some of the concepts: template type, template class, etc. The set of instances of **t** contains both the set of instantiations of **t** and the sets of all instantiations of subtypes of **t**. The sets of instantiations of different templates are always disjoint.