INTERNATIONAL STANDARD

ISO/IEC 10746-2

Second edition
2009-12-15

Information technology — Open distributed processing — Reference model: Foundations

Technologies de l'information — Traitement réparti ouvert — Modèle de référence: Fondements

Reference number
ISO/IEC 10746-2:2009(E)

Published by ISO in 2010

Published in Switzerland

CONTENTS

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 10746-2 was prepared 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 Rec. ITU-T X.902 (10/2009).

This second edition cancels and replaces the first edition (ISO/IEC 10746-2:1996), which has been technically revised.

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), Recommendations ITU-T X.901 | ISO/IEC 10746-1 to X.904 | ISO/IEC 10746-4, 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:

– Recommendation ITU-T 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 Rec. ITU-T X.903 | ISO/IEC 10746-3. This part is not normative.

– Recommendation ITU-T 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 Rec. ITU-T X.903 | ISO/IEC 10746-3 and to establish requirements for new specification techniques. This part is normative.

– Recommendation ITU-T 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 Rec. ITU-T X.902 | ISO/IEC 10746-2. This part is normative.

– Recommendation ITU-T 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 Recommendation | International Standard covers the concepts which are needed to perform the modelling of ODP systems (see clauses 6 to 14), and the principles of conformance to ODP systems (see 15).

The concepts defined in clauses 6 to 14 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.

Clause 15 defines how the various standards supported constrain an implementation and how such an implementation can be tested.

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

– Recommendation ITU-T 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 **data**: The representations of informationdealt with by information systems and users thereof.

3.2.2 **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.

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

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

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

3.2.6 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 some forms of representation to make it communicable, it is the interpretation of this representation (the meaning) that is relevant in the first place.

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.

3.2.8 viewpoint correspondence: A statement that some terms or other linguistic constructs in a specification from one viewpoint are associated with (e.g., describe the same entities as) terms or constructs in a specification from a second viewpoint. The forms of association that can be expressed will depend on the specification technique used.

NOTE – The terms associated by a correspondence need not necessarily be expressed using a single specification technique. The correspondence may associate a term in one specification technique with a term in some different specification technique. Rather than linking every individual pair of terms, general correspondences can also be expressed between specification techniques themselves. For example, composition operators defined in different specification techniques can be associated, implying correspondences wherever these operators are used to link terms in the respective viewpoints.

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 specification of 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.

Recommendation ITU-T 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, or 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, the language 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 it links.

7.3 **model**: A system of postulates, value declarations and inference rules presented as a description of a state of affairs (universe of discourse).

NOTE – Construction of a model allows precise description and reasoning about the state of affairs.

7.4 **specification**: A concrete representation of a model in some notation. Being in the real world, a specification can be inspected, manipulated or communicated.

NOTE 1 – The specification may itself be an entity in the universe of discourse of the model it represents, but in simple cases it will generally only be modelled in a separate universe of discourse addressing the system development process.

NOTE 2 – The specification can be instantiated by one or more implementations, particularly, for example, in the specification of commodity software products. Each instantiation of the specification will, in general, represent a separate universe of discourse and so lead to a separate set of entities with the relationships defined in the specification. Thus declaration of, for example, a singleton object (such as the ODP system) in a specification will lead to a separate ODP system instance each time the specification is implemented. This specification-instantiation distinction should be distinguished from the familiar type-instance distinctions between terms within the specification.

NOTE 3 – The relationship between a specification and its implementation underlies the conformance architecture defined in clause 15.

7.5 **notation**: A means of concrete representation for a particular type of a model, expressed as a grammar and suitable glyphs for its terminal symbols.

NOTE – One notation may be capable of representing a number of types of models, or of representing a specific viewpoint on a more general model.

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 8.7) and, dually, by its state (see 8.8). 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 8.12).

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 (see 13.3.1)). For modelling purposes, these functions and services are specified in terms of the behaviour of the object and of its interfaces (see 8.5). An object can perform more than one function. A function can be performed by the cooperation of several objects.

NOTE – 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.

NOTE 1 – "Action" means "action occurrence" not "action type". That is to say, different actions within a specification may be of the same type but still distinguishable in a series of observations. Depending on context, a specification may express that an action has occurred, is occurring or may occur.

This usage of action occurrence needs to be seen in the light of the notes on specification in 7.4. Thus the specification of a firework may require it to produce five flashes and a bang, which are six actions where flash and bang are action types. However, each member of a box of fireworks conforming to this specification will produce its own copy of this behaviour.

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

NOTE 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.

NOTE 4 – An object may interact with itself, in which case it is considered to play at least two roles in the interaction.

NOTE 5 – Involvement of the environment represents observability. Thus, interactions are observable whereas internal actions are not observable, because of object encapsulation. In most specification techniques observability is an implicit property of the environment and, therefore, it is not necessary to model the observer explicitly; however, there may, in some circumstances, be a need to include an explicit observer object in the specification, thereby increasing the cardinality of all interactions.

NOTE 6 – Observability of an action may depend on the level of specification. For instance, an action specification at one level of abstraction or in one viewpoint may correspond to a specification of multiple concurrent actions at a different level of abstraction or in another viewpoint. For example, a basic single function of a system in one viewpoint may be realized by multiple concurrent actions in a different viewpoint, defining a grid computing or sensor network, each one executing at the same time on network-connected computers in different locations. In this case, the observability of the occurrence of the basic single action can be deduced from the observability of those other multiple concurrent actions.

8.4 **event**: The fact that an action has taken place. When an event occurs, the information about the action that has taken place becomes part of the state of the system and may thus subsequently be communicated in other interactions. Such a communication is called an event notification; it carries the information about the event from the object that performs or observes it to other objects that have a need to take action as a result of it.

NOTE 1 – An action changes the state of the objects participating in it; an event is the fact that the action has occurred; an event notification is a communication about the event, caused by some action; the receipt of the notification changes the state of objects not participating in the original action.

NOTE 2 – An event notification may convey information about the fact that an internal action has occurred. For example, an internal action may change the availability of some server and a subsequent event notification may convey this fact to its potential clients.

8.5 **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.

NOTE 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.

NOTE 2 – The phrase "an interface between objects" is used to refer to the binding (see 13.5.2) between interfaces of the objects concerned. In the two-party case, such bindings normally link interfaces with complementary causalities. For example, in a client-server binding (see 13.4.5 and 13.4.6), a client initiating interface is bound to a server providing interface. In many specification languages, the fact that the client has an initiating interface is not explicit, but is indicated by stating a requirement for the kind of server needed if the client is to operate successfully, i.e., the concept of a required interface.

NOTE 3 – An interface of an object may be used by other objects. Using interfaces provided by other objects may constitute a part of the object's behaviour.

NOTE 4 – If an interface is provided by an object, part of the providing object's behaviour is triggered when this interface is used by other objects. If an object uses an interface of some providing object, this is expressed by its behaviour involving an interaction which forms part of its own initiating interface. The interaction in the first object's initiating interface is associated with the corresponding interaction in the other object's providing interface as a result of the binding process between the two interfaces. An object may provide both initiating and providing interfaces.

8.6 **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).

8.7 **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.

NOTE 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.

NOTE 2 – Action and activity are degenerate cases of behaviour.

NOTE 3 – In general, several sequences of interactions, called traces (see 9.7) are consistent with a given behaviour.

8.8 **state (of an object)**: At a given instant in time, the condition of an object that determines the set of all sequences of actions (or traces) in which the object can participate.

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.

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

NOTE 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.

NOTE 2 – Every interaction is an instance of a communication.

NOTE 3 – Any communication can be seen as an interaction by abstracting away intermediate objects involved in the communication.

NOTE 4 – Any communication (and, hence, any interaction) can be provided by a wide range of technologies such as remote invocation, message transfer, etc.

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

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

NOTE 1 – The extent of the interval in time or space is chosen to reflect the requirements of a particular modelling task and the properties of a particular specification technique. 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.

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

8.12 **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.

8.13 **relation**: An association between two or more domains of entities. In RM-ODP, relations can be defined for, at least, objects, interfaces and actions.

8.14 **relationship**: An association between two or more entities.

NOTE – Relationships are instances of relations.

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.

NOTE 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.

NOTE 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 represent dual specification.

9.4 **behavioural compatibility**: An object is behaviourally compatible with a second object with respect to a set of criteria (see Notes 1 and 2) 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).

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