ISO/IEC JTC1/SC7
Software & System Engineering
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC7 N2565

2002-01-11

Document Type	FDIS Ballot
Title	FDIS 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise language.
Source	JTC1/SC7 Secretariat
Project	07.77
Status	Sent to ITTF for FDIS balloting.
References	N2564
Action ID	FYI or ACT
Due Date	
Mailing Date	2002-01-11
Distribution	SC7_AG
Medium	Encoded Acrobat
No. Of Pages	26
Note	This docujment is circulated to SC7 member bodies FOR INFORMATION PURPOSE ONLY. The document has been sent to ITTF for FDIS balloting. This is NOT a SC7 ballot.

Address reply to: ISO/IEC JTC1/SC7 Secretariat
Bell Canada – Office of the CIO
1050 Beaver Hall Hill, 2nd Floor, Montréal (Québec) Canada H2Z 1S4
Tel.: +1 (514) 391-8286 Fax: +1 (514) 870-2246
sc7@qc.bell.ca

Form 10 - electronic

EXPLANATORY REPORT ISO/IEC JTC 1/SC 7 **N2565**	**ISO/IEC FDIS 15414**
Will supersede: SC7 N2359	Secretariat: Standards Council of Canada (SCC)

This form should be sent to ITTF, together with the committee draft, by the secretariat of the joint technical committee or sub-committee concerned.

The accompanying document is submitted for circulation to member body vote as a FDIS, following consensus of the P-members of the committee obtained on: 2001-01-09	
X	By email ballot initiated on: 2000-09-09

P-members in favor:	10
P-members voting against:	5
P-members abstaining:	1
P-members who did not vote:	10
O-members in favor	1

Remarks:

This document is ready for FDIS balloting.

Project: 07.77

I hereby confirm that this draft meets the requirements of part 3 of the IEC/ISO Directives.

Date: 2002-01-11	Name and signature of the secretary: Jean-Normand Drouin

INTERNATIONAL STANDARD 15414

ITU-T RECOMMENDATION X.911

INFORMATION TECHNOLOGY – OPEN DISTRIBUTED PROCESSING – REFERENCE MODEL – ENTERPRISE LANGUAGE

CONTENTS

Introduction

The rapid growth of distributed processing has led to the adoption of the Reference Model of Open Distributed Processing (RM-ODP). This Reference Model provides a co-ordinating framework for the standardisation of open distributed processing (ODP). It 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.

The Reference Model of Open Distributed Processing 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.

This Recommendation | International Standard refines and extends the definition of how ODP systems are specified from the enterprise viewpoint, and is intended for the development or use of enterprise specifications of ODP systems.

0.1 RM-ODP

The RM-ODP consists of:

- Part 1: ITU-T Recommendation 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 categorisation of required areas of standardisation expressed in terms of the reference points for conformance identified in ITU-T Recommendation X.903 | ISO/IEC 10746-3. This part is not normative.

- Part 2: ITU-T Recommendation X.902 | ISO/IEC 10746-2: **Foundations**: which contains the definition of the concepts and analytical framework for normalised 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 Recommendation X.903 | ISO/IEC 10746-3 and to establish requirements for new specification techniques. This part is normative.

- Part 3: ITU-T Recommendation 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 must conform. It uses the descriptive techniques from ITU-T Recommendation X.902 | ISO/IEC 10746-2. This part is normative.

- Part 4: ITU-T Recommendation X.904 | ISO/IEC 10746-4: **Architectural semantics**: which contains a formalisation of the ODP modelling concepts defined in ITU-T Recommendation X.902 | ISO/IEC 10746-2 clauses 8 and 9. The formalisation is achieved by interpreting each concept in terms of the constructs of one or more of the different standardised formal description techniques. This part is normative.

- ITU-T Recommendation X.911 | ISO/IEC 15414: **Enterprise language**: this Recommendation | International Standard.

0.2 This Recommendation | International Standard

Part 3 of the Reference Model, ITU-T Recommendation X.903 | ISO/IEC 10746-3, defines a framework for the specification of ODP systems comprising

1) five viewpoints, called enterprise, information, computational, engineering and technology, which provide a basis for the specification of ODP systems;
2) a viewpoint language for each viewpoint, defining concepts and rules for specifying ODP systems from the corresponding viewpoint.

The purpose of this Recommendation | International Standard is to:

--Refine and extend the enterprise language defined in ITU-T Recommendation X.903 |ISO/IEC 10746-3 to enable full enterprise viewpoint specification of an ODP system;

--Explain the correspondences of an enterprise viewpoint specification of an ODP system to other viewpoint specifications of that system; and

--Ensure that the enterprise language when used together with the other viewpoint languages is suitable for the specification of a concrete application architecture to fill a specific business need.

This ITU-T Recommendation X.911 | ISO/IEC IS 15414 uses concepts taken from ITU-T Recommendations X.902 and X.903 | ISO/IEC 10746-2 and 10746-3 and structuring rules taken from Clause 5 of ITU-T Recommendation X.903 | ISO/IEC 10746-3; it introduces refinements of those concepts, additional viewpoint-specific concepts, and prescriptive structuring rules for enterprise viewpoint specifications. The additional viewpoint-specific concepts are defined using concepts from ITU-T Recommendations X.902 and X.903 | ISO/IEC 10746-2 and 10746-3.

0.3 Overview and Motivation

The purpose of this Recommendation | International Standard is to provide a common language (a set of terms and structuring rules) to be used in the preparation of an enterprise specification capturing the purpose, scope and policies for an ODP system. Such an enterprise specification forms part of the specification of an ODP system in terms of the set of viewpoints defined by ITU-T Recommendation X.903 | ISO/IEC 10746-3. The primary audience for this document consists of those who prepare and use such specifications.

An enterprise specification is part of an ODP system specification. It can describe any or all of:

- an existing system;
- an anticipated future structure or behaviour of that existing system
- a system to be created within some environment

The motivation for a standard enterprise language is to support standardized techniques for specification in order to improve communication and help create specifications that are consistent overall. The enterprise language provides the terms and structuring rules to specify the purpose, scope and policies for an ODP system in a manner that is meaningful for the stakeholders for that system, including the owners, the users, the developers and the maintainers.

An enterprise specification describes the structure and behaviour of the system within its environment. It explicitly includes those aspects of the environment that influence the behaviour of the ODP system – environmental constraints are captured as well as usage and management rules. Policies about potential changes in the system that may rule its future evolution may also be included. Such an environment can be a technical environment (e.g., the software and hardware environment of a service component) or a social or business organisation (e.g., a group of co-operating companies, a particular service inside a company).

When preparing a specification, there are many approaches that are used for understanding, reaching agreement about, and specifying systems in the context of the organizations of which they form a part. Many of these approaches fall into the categories often referred to as analysis or requirements specification. They can provide useful insights into both the organization under consideration and the requirements for systems to support it, but they often lack the rigour, consistency and completeness needed for thorough specification. It is a key objective of this Recommendation | International Standard to provide a way of relating the commonly used concepts and underlying principles of such approaches to the modelling framework of the RM-ODP.

An important objective of an enterprise specification is to support an agreement (for example, as part of the contract for the supply of a system) between the potential owners and users of an ODP system and the provider of that system. Both parties should be able to write, read and discuss such a specification, the owners and users to be sure of the expected structure and behaviour of the system that they will get, and the provider to be clear about the structure and behaviour of the system being provided.

Enterprise specifications can also be used at other phases of the system life-cycle. The specification can, for example, be used at system run-time to control agreements between the system and its users, to establish new agreements according to the same contract structure and to establish federations.

Finally, in the context of the current trend to integrate existing systems into global networks, where the functionality of interest spans multiple organizations, the enterprise language provides means to specify the joint agreement regarding the structure and behaviour of the ODP systems within and between these organizations.

The concepts and structuring rules this document provides may be used for development of software engineering methodologies and tools exploiting ODP viewpoint languages, and for development of textual or graphical notations for ODP enterprise language itself. For these purposes, this document provides rules for the information content of an enterprise specification and the grouping of that information. Further requirements on the relationships between enterprise language concepts and their correspondences to concepts in other viewpoints are specific to the methodologies, tools or notations to be developed.

INTERNATIONAL STANDARD 15414

ITU-T RECOMMENDATION X.911

INFORMATION TECHNOLOGY– OPEN DISTRIBUTED PROCESSING – REFERENCE MODEL – ENTERPRISE LANGUAGE

1 Scope

This Recommendation | International Standard provides:

a) a language (the enterprise language) comprising concepts, structures, and rules for developing, representing, and reasoning about a specification of an ODP system from the enterprise viewpoint (as defined in ITU-T Recommendation X.903 | ISO/IEC 10746-3);

b) rules which establish correspondences between the enterprise language and the other viewpoint languages (defined in ITU-T Recommendation X.903 | ISO/IEC 10746-3) to ensure the overall consistency of a specification.

The language is specified to a level of detail sufficient to enable the determination of the compliance of any modelling language to this Recommendation | International Standard and to establish requirements for new specification techniques.

This Recommendation | International Standard is a refinement and extension of ITU-T Recommendation X.903 | ISO/IEC 10746-3, clauses 5 and 10, but does not replace them.

This Recommendation | International Standard is intended for use in preparing enterprise viewpoint specifications of ODP systems, and in developing notations and tools to support such specifications.

As specified in clause 5 of ITU-T Recommendation X.903 | ISO/IEC 10746-3, an enterprise viewpoint specification defines the purpose, scope and policies of an ODP system. [see also 3-5.0]

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 the currently valid ITU-T Recommendations.

Identical ITU-T Recommendations | International Standards

– ITU-T Recommendation X.902 (1995) | ISO/IEC 10746-2: 1994, *Information technology – Open Distributed Processing – Reference Model – Foundations*

– ITU-T Recommendation X.903 (1995) | ISO/IEC 10746-3: 1994, *Information technology – Open Distributed Processing – Reference Model – Architecture*

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

3 Definitions

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.

-- action;

-- behaviour (of an object) ;

-- composite object;

-- composition;

-- configuration (of objects);

-- conformance;

-- conformance point;

-- contract;

-- <X> domain;

-- entity;

-- environment contract;

-- environment (of an object);

-- epoch;

-- establishing behaviour;

-- instance (of a type);

-- instantiation (of an <X> template);

-- interface;

-- internal action;

-- invariant;

-- liaison;

-- location in time;

-- object;

-- obligation;

-- ODP standards;

-- ODP system;

-- permission;

-- prohibition;

-- proposition;

-- reference point;

-- refinement;

-- role;

-- state (of an object) ;

-- subtype;

-- system;

-- <X> template;

-- terminating behaviour;

-- 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;

-- channel

-- community;

-- computational behaviour;

-- computational binding object;

-- computational object;

-- computational interface;

-- computational viewpoint;

-- correspondence;

-- dynamic schema;

-- engineering viewpoint;

-- enterprise viewpoint;

-- <X> federation

-- information object;

-- information viewpoint;

-- interceptor;

-- invariant schema;

-- node;

-- protocol object;

-- static schema;

-- stub;

-- technology viewpoint.

-- <viewpoint> language;

3.2 Definitions from ODP standards extended in this standard

This Recommendation | International Standard extends the definition of the following term originally defined in ITU-T Rec. X.902 | ISO/IEC 10746-2. [2-11.2.7] :

-- policy

The extended definition is in clause 6 of this document.

4 Abbreviations

ODP	open distributed processing
RM-ODP	Reference Model of Open Distributed Processing (ITU-T Recommendations X.901 to X.904 | ISO/IEC 10746 Parts 1-4)

5 Conventions

This document contains references to Parts 2 and 3 of RM-ODP. For example, [2-9.4] is a reference to Part 2, (ITU-T Rec. X.902 | ISO/IEC 10746-2), subclause 9.4. These references are for the convenience of the reader.

This document also contains some text which is a modification of text from Part 3, ITU-T Rec. X.903 | ISO/IEC 10746-3. Such text is marked by a reference like this: [see also 3-5.0]. The modifications are authoritative with respect to the enterprise language.

6 Concepts

The concepts of the enterprise language defined in this document comprise:

--the concepts identified in 3.1.1 and 3.1.2 as they are defined in ITU-T Rec. X.902 | ISO/IEC 10746-2 and in ITU-T X.903 | ISO/IEC 10746-3;

--the concepts defined in this clause.

This clause defines new concepts and refines the definition of policy from ITU-T Rec. X.902 | ISO/IEC 10746-2. [2-11.2.7] The grouping into subclauses and the headings of the subclauses of this clause are not normative.

6.1 System concepts

6.1.1 Scope (of a system): The behaviour that system is expected to exhibit.

6.1.2 Field of application (of a specification): the properties the environment of the ODP system must have for the specification of that system to be used.

6.2 Community concepts

6.2.1 Objective: Practical advantage or intended effect, expressed as preferences about future states.

NOTES:

1 – Some objectives are ongoing, some are achieved once met.

2 – In the text of ITU-T Rec. X.903 | ISO/IEC 10746-3 [3-5] the terms, purpose and objective, are synonymous. The enterprise language systematically uses the term, objective, and emphasises the need of expressing objective in measurable terms.

6.2.2 Community object: A composite enterprise object that represents a community. Components of a community object are objects of the community represented.

6.3 Behaviour concepts

6.3.1 Actor (with respect to an action): An enterprise object that participates in the action.

NOTE - It may be of interest to specify which actor initiate that action.

6.3.2 Artefact (with respect to an action): An enterprise object that is referenced in the action.

NOTE – An enterprise object that is an artefact in one action can be an actor in another action.

6.3.3 Resource: An enterprise object which is essential to some behaviour and which requires allocation or may become unavailable.

NOTES

1 – Allocation of a resource may constrain other behaviours for which that resource is essential.

2 – A consumable resource may become unavailable after some amount of use or after some amount of time (in case a duration or expiry has been specified for the resource)..

6.3.4 Interface role: A role of a community identifying behaviour which takes place with the participation of objects that are not a members of that community.

6.3.5 Process: A collection of steps taking place in a prescribed manner and leading to an objective.

NOTES

1 – A process may have multiple starting points and multiple end points.

2 – The prescribed manner may be a partially ordered sequence.

3 – A process specification can be a workflow specification.

4 – The activity structure concepts provided in subclause 13.1 of ITU-T Rec. X.902 | ISO/IEC 10746-2 may be used, after substitution of ‘step’ for ‘action’ and ‘process’ for ‘activity,’ to specify the structure of a process.

5 – An enterprise specification may define types of processes and may define process templates.

6.3.6 Step: An abstraction of an action, used in a process, that may leave unspecified objects that participate in that action.

6.4 Policy concepts

6.4.1 Policy: A set of rules related to a particular purpose. A rule can be expressed as an obligation, an authorization, a permission or a prohibition.

NOTES

1 – Not every policy is a constraint. Some policies represent an empowerment.

2 – This definition refines 2-11.2.7 by adding authorization.

6.4.2 Authorisation: A prescription that a particular behaviour must not be prevented.

NOTE – Unlike a permission, an authorisation is an empowerment

6.4.3 Violation: An action contrary to a rule.

NOTE – A rule or policy may provide behaviour to occur upon violation of that or some other rule or policy.

6.5 Accountability concepts

6.5.1 Party: An enterprise object modelling a natural person or any other entity considered to have some of the rights, powers and duties of a natural person.

NOTES

1 – Examples of parties include enterprise objects representing natural persons, legal entities, governments and their parts, and other associations or groups of natural persons.

2 – Parties are responsible for their actions and the actions of their agents.

The following concepts are used to identify actions which involve the accountability of a party.

6.5.2 Commitment: An action resulting in an obligation by one or more of the participants in the act to comply with a rule or perform a contract.

NOTE – The enterprise object(s) participating in an action of commitment may be parties or agents acting on behalf of a party or parties. In the case of an action of commitment by an agent, the principal becomes obligated.

6.5.3 Declaration: An action that establishes a state of affairs in the environment of the object making the declaration.

NOTE – The essence of a declaration is that, by virtue of the act of declaration itself and the authority of the object or its principal, it causes a state of affairs to come into existence outside the object making the declaration.

6.5.4 Delegation: The action that assigns authority, responsibility or a function to another object.

NOTE – A delegation, once made, may later be withdrawn.

6.5.5 Evaluation: An action that assesses the value of something.

NOTES

1 – For example, the act by which an ODP system assigns a relative status to some thing, according to estimation by the system.

2 – Value can be considered in terms of usefulness, importance, preference, acceptability etc; the evaluated target may be, for example, a credit rating, a system state, a potential behaviour, etc.

6.5.6 Prescription: An act that establishes a rule.

6.5.7 Agent: An enterprise object that has been delegated (authority, responsibility, a function, etc.) by and acts for another enterprise object (in exercising the authority, carrying out the responsibility, performing the function, etc.).

NOTES

1 – An agent may be a party or may be the ODP system or one of its components. Another system in the environment of the ODP system may also be an agent.

2 – The delegation may have been direct, by a party, or indirect, by an agent of the party having authorisation from the party to so delegate.

6.5.8 Principal: A party that has delegated (authority, a function, etc.) to another.

6.5.9 Contracting party (with respect to a contract): A party that agrees to that contract.

7 Structuring Rules

This clause refines and extends the structuring rules defined in subclause 5.2 of ITU-T Rec. X.903 | ISO/IEC 10746-3, as they apply to the concepts of community, enterprise object, objective, behaviour and policy. It defines structuring rules for the accountability concepts defined in subclause 6.5 of this document. It uses the concepts defined in ITU-T Rec. X.902 | ISO/IEC 10746-2, in subclause 5.1 of ITU-T Rec. X.903 | ISO/IEC 10746-3 and in clause 6 of this document.

7.1 Overall structure of an enterprise specification

An enterprise specification of an ODP system is a description of that system and relevant parts of its environment. The enterprise specification focuses on the scope and purpose of that system and the policies that apply to it in the context of its environment.

NOTE - The environment of an ODP system and the ODP system itself may span multiple organizations.

A fundamental structuring concept for enterprise specifications is that of community. A community is a configuration of enterprise objects that describes a collection of entities (e.g. human beings, information processing systems, resources of various kinds and collections of these) that is formed to meet an objective. These entities are subject to an agreement governing their collective behaviour. The assignment of actions to the enterprise objects that comprise a community is defined in terms of roles. (see 7.8.1 and 7.8.2)

The ODP system may play a role in more than one community. Thus, the enterprise specification describes, within the areas of interest of the specification users:

-- roles fulfilled by the ODP system and enterprise objects in its environment;

-- steps within processes in which the ODP system and enterprise objects in its environment participate;

-- policy for the system, including those of environment contracts.

An enterprise specification of an ODP system includes at least the community in which that system may be represented as a single enterprise object interacting with its environment. Whether the specification actually includes that level of abstraction is left for the specifier to decide.

NOTE – This minimal enterprise specification describes the objective and scope of the ODP system; this description is necessary for completeness of the enterprise specification.

Where necessary for clarity or completeness of description of the behaviour of the ODP system, the enterprise specification can include any other communities of which the ODP system or its components are members, and other communities of which enterprise objects in the environment of the ODP system are members.

NOTE – The set of communities in an enterprise specification may include, for example, communities at both more abstract and more detailed levels than the minimal enterprise specification, as well as communities relating to functional decomposition of the ODP system and to ownership of the ODP system and its parts.

The enterprise specification can also be structured in terms of a number of communities interacting with each other.

NOTE – Such structuring may represent, for example, a federation.

The interaction of communities may be represented in several ways:

-- The communities concerned may be viewed as composite objects (c-objects), and a new community formed of those composite objects. That community may be described explicitly or may be left implicit.

-- A new community may be formed of one or more objects of each community, with roles in the new community specifying interactions of the communities.

-- A new community may be formed of one or more objects of each community, by an assignment of roles in different communities to the same object, and a correspondence of each of those roles to a role in the new community.

-- Interface objects of the communities may interact, forming a new community.

The scope of the system is defined in terms of its intended behaviour; in the enterprise language this is expressed in terms of roles or processes or both, policies and the relationships of these.

NOTE - It may be meaningful to discuss the intended, delivered or expected scope of a system in various phases of planning, development or deployment. When so, the term "scope" should be appropriately qualified.

A complete ODP system specification indicates rules for internal consistency in terms of relationships between various viewpoint specifications and a complete enterprise specification contains conformance rules that define the required behaviour of the described ODP system

This clause 7 defines how the concepts identified in clause 3 or defined in clause 6 of this document are used in an enterprise specification.

7.2 Contents of an enterprise specification

An enterprise specification is structured in terms of the elements explained in subclause 7.1 (communities, enterprise objects, contracts, roles, interactions, activities, behaviour, the ODP system and policies) and the other concepts identified in Clause 10, as well as the relationships between them.

For each of these elements, depending on the specifier's choice and desired level of detail, the enterprise specification provides:

-- the characteristics of the element, or

-- the type or types of the element, or

-- a template for the element.

An enterprise specification provides a pattern for realisation of an ODP system in its environment. As such it may be realised once, never, or many times, depending upon the objective of the specifier. This means that the behaviour defined may also be observable any number of times, depending on when and where the specification is realised. It is therefore necessary to take care of the context when interpreting statements about the occurrence of the concepts in an enterprise specification .

In particular, when distinguishing type and occurrence in a specification, the objective is normally to distinguish between multiple occurrences of a single type within the specification, and not to imply a constraint on how often the specification can be realized in the world. The definitions in this document should be interpreted in the context of specification, without constraining when and where the specification should be realised.

The enterprise language makes no prescription about the specification process nor about the level of abstraction to be used in an enterprise specification.

NOTES

1 – No recommendations are made about the relative merits of modelling from top-down or bottom-up. Nor is there a recommended sequencing of the development of viewpoint specifications.

2 – It is a design choice whether a specification deals with a specific implementation by, for example, identifying individual enterprise objects, or deals with a more flexible architecture by identifying types and rules for assigning enterprise objects to roles.

3 – A specification may be partitioned because of readability, reuse of specification fragments in other specifications or interoperability of enterprise objects.

4 - Roles and communities, as well as types and templates, can be private to a specification and development environment, or they can be stored in a repository that can be shared by a wider audience of several development environments and groups.

7.3 Community rules

7.3.1 Community

An enterprise specification states the objective of a community, how it is structured, what it does, and what objects comprise it. The objective of the community is expressed in a contract that specifies how the objective can be met. This contract:

- states the objective for which the community exists,
- governs the structure, the behaviour and the policies of the community,
- constrains the behaviour of the members of the community,
- states the rules for the assignment of enterprise objects to roles.

The contract for the community specifies constraints that govern the existence or behaviour of the collection of entities described by the community. When a collection of entities is represented as a community, there may already be some implicit or explicit agreement about those entities. Terms of that agreement may appear in the community contract. An enterprise specification may include all or part of that agreement by reference. Such references relate the elements of the specification to terms of that agreement. In particular, commitments of enterprise objects may be subject to that agreement.

The behaviour of the community is such that meets its objective. The objects of a community are constrained by the rules of the community contract.

The contract can be put in place by a defined behaviour carried out by enterprise objects or the contract may be prescribed to exist by the enterprise specification.

The collective behaviour of the community is specified in terms of one or more of the following elements:

- the roles of the community (including those roles which define how a community interacts with its environment),
- the processes that take place in the community,
- the assignment of roles to steps in processes,
- policies that apply to the roles and processes and
- identification of those actions for which parties are accountable.

This collective behaviour is constrained by the policies associated with roles and processes and by the community contract.

The behaviours of objects in a community are subject to the contract of that community and to the constraints specified in relationships between those objects.

The structure of the community is defined in terms of the following elements:

- roles;
- policies for assignment of enterprise objects to roles;
- relationships between roles;
- relationships of roles to processes;
- policies that apply to roles and to relationships between roles;
- policies that apply to relationships between enterprise objects in the community;
- behaviour that changes the structure or the members of the community during the lifetime of that community.

NOTES

1 – Types of communities or a community template may be used in the specification of a community.

2 – Types of communities may be related by refinement.

3 – A family of related contracts may be generated from a contract template. Some aspects of the contract (e.g. membership) may only apply to particular instantiations of the contract template, while other aspects may apply to all instantiations of the contract template. For example, assignment rules and policies can be considered as parameters in a contract template. The style of contract specification determines the method of community establishment, as well as other aspects of the community life-cycle.

7.3.2 Relationships between communities

An enterprise specification can include one or more communities. Interactions between enterprise objects fulfilling appropriate roles within different communities can be considered as interactions between those communities.

Communities may interact in the following ways:

-- a community object fulfils one or more roles in other communities

-- two or more community objects interact in fulfilling roles in another some other community

-- the enterprise specification requires the same object is required to fulfil specific roles in more than one community

-- a community includes behaviour for creating new communities

NOTES

1 – For example, federation establishment means creation of a new community involving the definition of an appropriate policy framework, the structure for that community and the community contract.

2 – For interactions involving community objects and the communities they represent see 7.8.3 – Interface Roles

For each of these ways of interacting there is an invariant that determines the constraints on the collective behaviour of the communities concerned.

These invariants include:

-- where a community object fulfils one or more roles in another other community, the community that the community object represents is governed by the policies of the other community

-- where two or more community objects interact in fulfilling roles in another some other community, the communities that the community objects represent are related by those interactions

-- where the same object is required to fill specific roles in more than one community, an invariant specifies how the actions of that object affect those communities

-- where the same object is required to fill specific roles in more than one community, that object becomes governed by the policies of all the communities.

-- where two or more communities interact, there is a set of policies common two to those communities

NOTES

1 – Where two communities interact an implicit community may be considered, such that the community objects representing both communities are members of and are governed by the policies of that community. The element of shared objective and the common set of policies can be formed either at design time and included in the specifications of the communities or left for run-time negotiation or testing of acceptability during community population.

2 – The communities involved may have differing policy rules to all of which the enterprise object should be able to conform.

7.4 Enterprise object rules

An enterprise specification will include enterprise objects; an enterprise object is any object in an enterprise specification. Any enterprise objects and the entities they model are those felt to be necessary or desirable to specify the system from the enterprise viewpoint or to understand the enterprise specification.

NOTE – An enterprise object may be a model of a human being, a legal entity, an information processing system, a resource or a collection or part of any of these.

An enterprise object may be refined as a community at a greater level of detail. Such an object is then a community object.

All enterprise objects in an enterprise specification fulfil at least one role in at least one community. In fulfilling their roles, enterprise objects, participate in actions, some of which are interactions with other enterprise objects. The behaviour of an enterprise object is restricted by the roles to which it is assigned.

An enterprise object may be a member of a community because:

-- by design the community includes the object,

-- the object becomes a member of the community at the time of creation of that community, or

-- the object becomes a member of the community as a result of dynamic changes in the configuration of the community.

NOTES:

1 – The community contract includes rules for the assignment of enterprise objects to roles; thus, to establish a community it is not necessary to identify the enterprise objects of that community.

2 – The community contract can include rules that change the community structure (for example, the number of roles).

7.5 Common community types

Two community types are:

-- <X>-domain

-- <X>-federation

Communities of these types can be specified so that they overlap totally or partially. These basic community types do not imply any hierarchical relationships. A specification may choose to use some or none of these community types.

7.5.1 <X>-domain community type

An<X>-domain community comprises an <X>-domain of enterprise objects in the roles of controlled objects and an enterprise object in the role of controlling object for the <X>-domain. The <X>-domain community establishes the characterizing relationship <X> between the enterprise objects in the roles of controlled objects and the enterprise object in the role of controlling object.

7.5.2 <X>-federation community type

An <X>-federation community contains two or more <X>-federation member roles which are filled by <X>-domain communities. The objective of an <X>-federation is to enable the characterizing relationship, <X>, between controlled objects and controlling object in each <X>-domain community to be shared with the controlling objects of the other <X>-domain communities. The specific manner in which the characterising relationship, <X>, is shared requires further refinement of the federation community type. Controlling objects are subject to the policies of the contract of the <X>-federation community and may commit their respective <X>-domain communities to some contract, but may prescribe policies ~~on~~ for each other's <X>-domain community only as provided in the contract of the <X>-federation community.

Note. At the level of abstraction at which federation is agreed, the federation members must be <X>-domain<X><X> communities having the same characterizing relationship, <X>. However, each <X>-domain community may actually be an instance of one or more refined <X>-domain community types.

7.6 Lifecycle of a community

7.6.1 Establishing a community

An enterprise specification can include establishing behaviour for a community.

The establishing behaviour may be implicit or explicit, but it establishes the required structures and responsibilities to maintain and control the community, for example, the policy framework, the community contract and the objects in the community. Objects of the community may need to be instantiated as a part of the establishing behaviour.

7.6.2 Assignment Rules

The establishing behaviour by which a community is established includes the assignment of enterprise objects to roles. The community contract specifies assignment rules for choosing enterprise objects to fulfil the specified roles. The enabled behaviour is consistent with the roles.

NOTES

1 – The role/object relationship is not a type/instance relationship.

2 – The assignment process can be late and dynamic, i.e., a role can be fulfilled by a enterprise object through a match-making process that considers, with respect to the requirements stated for the role, the interfaces and behaviour of that object, and, in the case of a community object, the policies of the community it represents.

Members to the community can be selected on demand according to the assignment rules for that community.

The assignment rules can directly identify the objects, or may use a supporting mechanism using more complex assignment rules. Such rules may be based on object identifiers, relationships between objects, object capabilities, technologies, preceding commitments, object behaviour, etc.

7.6.3 Changes in a community

Changes in the structure or behaviour of a community can occur only if an enterprise specification includes behaviour that can cause such changes.

The changes to be considered here include:

-- introduction of new rules within the existing policy framework; or changing the existing rules; and

-- introduction of new roles into the community.

The enterprise objects assigned to roles in the community can be dynamically changed during the lifetime of the community. As a consequence, a role can, subject to other constraints, have no enterprise object assigned to it. Still, the community is continuously responsible for the obligations placed on that role.

If an enterprise object ceases to fulfil the assignment rule associated to it, it violates the community contract.

7.6.4 Terminating a community

An enterprise specification can include terminating behaviour for a community; it must do so if it includes establishing behaviour for that community.

NOTES

1 – For example, a community contract may provide for termination when the objective is achieved. A violation may be associated with a recovery behaviour, which may be the termination of the community.

2 – Some communities are permanent and never terminate.

7.7 Objective rules

Every community has exactly one objective. The objective is expressed in a contract which specifies how the objective can be met. An objective can be a composition of sub-objectives.

An enterprise specification may decompose the objective of a community into sub-objectives. A sub-objective may be assigned to a collection of roles; in that case, the behaviour of the collection of roles is specified to meet the sub-objective and the sub-objective is met by the collection of objects performing the actions of the collection of roles.

A sub-objective may be assigned to a process; in that case, the process is specified to meet the sub-objective and the sub-objective is met by the actions of objects performing the process. In this case, the sub-objective defines the state in which the process terminates.

The policies of a community restrict the community behaviour in such a way that it is possible to meet the objective. Such policies result in behaviour that suits the objective of the community.

When a community-object fulfils a role in another community, the objective of the community of which the community-object is an abstraction is consistent with any sub-objectives assigned to that role in the other community.

NOTE - An enterprise specification may provide for detection of conflicts in objectives and for resolution of those conflicts.

7.8 Behaviour rules

7.8.1 Roles and processes

The behaviour of a community is a collective behaviour consisting of the actions in which the objects of the community participate in fulfilling the roles of the community, together with a set of constraints on when these actions may occur.

NOTE – There are many specification styles for expressing when actions may occur (e.g., sequencing, pre-conditions, partial ordering, etc.). The modelling language chosen for expressing an enterprise specification may impose certain styles.

The assignment of actions to the enterprise objects that comprise a community is defined in terms of roles. A role identifies an abstraction of the community behaviour. All of the actions of that role are associated with the same enterprise object in the community. Each action of the community is either part of a single role behaviour or is an interaction that is part of more than one role behaviour. Each of these abstractions is labelled as a role. The behaviour identified by that role is subject to the constraints specified in the contract and structure of the community. In contrast to the specification of actions and their ordering in terms of processes (see below), the emphasis is on the enterprise objects that participate in the particular behaviour.

Roles are used to decompose the behaviour of the community into parts that can each be performed by an enterprise object in the community. The enterprise object that performs the behaviour of a role is said to fulfil that role within the community or is said to be assigned to that role within the community.

Each action will be part of at least one role, but can be part of many role (when the action involves an interaction).

The actions and their ordering can be defined in terms of processes. A process identifies an abstraction of the community behaviour that includes only those actions that are related to achieving some particular result/purpose/sub-objective within the community. Each abstraction is labelled with a process name. In contrast to the specification of actions as related to roles (see above), the emphasis is on what the behaviour achieves.

Processes decompose the behaviour of the community into steps.

NOTE – The choice of using a role-based or process-based modelling approach will depend on the modelling method used and the aim of modelling. It may be necessary to use a combination of the two approaches.

7.8.2 Role rules

In a community contract, each role stands as a placeholder for some enterprise object that exhibits the behaviour identified by the role. For each role there is an assignment rule that sets requirements for objects that may fulfil that role.

An enterprise object may fulfil several roles in one community, and may fulfil roles in several communities. An object fulfilling several roles becomes constrained simultaneously by all the behaviours identified by those roles and by the policies that apply to those roles.

NOTE – If the term '<X> object' is used in an enterprise specification, where f<X> is a role, it should be interpreted as meaning 'an enterprise object fulfilling the role, <X>'. Where an enterprise object fulfils multiple roles, the names can be concatenated.

At any location in time at most one enterprise object fulfils each role. The constraints of the behaviour identified by the role become constraints on the object fulfilling the role. A role may be fulfilled by different objects at different times or be unfulfilled, provided that the specification of the community so permits.

An enterprise specification may include a number of roles of the same type each fulfilled by distinct enterprise objects, possibly with a constraint on the number of roles of that type that can occur.

NOTE - Examples are modelling the members of a committee and modelling the customers of a service.

An enterprise object assigned to a role must be of a type behaviourally compatible with that role, unless the specification includes mechanisms to determine and resolve any incompatibilities. [2-9.4]

Note – Enterprise specifications may refer to existing mechanisms for determining and resolving incompatibilities between types of objects and requirements set by roles, thus enlarging the set of objects acceptable for a given role.

An enterprise specification may allow roles and relationships between roles to be created or deleted during the lifetime of the community. The role lifetime is contained within the community lifetime, and the period for which a particular enterprise object fulfils a given role is contained within the lifetime of that role.

NOTE - The constraints of the community should be satisfied throughout its lifetime. However, these invariants may change; this may determine different epochs in this lifetime. Such changes may lead to changes in the sets of roles and in the sets of relationships between roles of the community.

An assignment policy is a set of rules of a community which govern the selection of an enterprise object to fulfil a role.

NOTE - The rules define what the object to fulfil a role must be capable of doing and not restricted from doing by earlier commitments, and what relationships to other objects are required or prohibited.

7.8.3 Interface roles and interactions between communities

One or more roles in a community may identify behaviour that includes interactions with objects outside that community; these are interface roles.

In such a case a community may be specified at two different levels of abstraction:

-- as a configuration of enterprise objects, where some of these objects fulfil interface roles, and

-- as a community object that is an abstraction of the community. Interactions in which the community object can participate as part of some other community are identified by the interface roles of the community that community object represents.

The behaviour identified by an interface role may include internal actions.

7.8.4 Enterprise objects and actions

An enterprise object fulfils at least one role in at least one community

A way of categorising the involvement of an enterprise object is:

-- The object can participate in carrying out the action; in this case it is said to be an actor with respect to that action.

-- The object can be mentioned in the action; in this case it is said to be an artefact with respect to that action.

-- The object can both be essential for the action and require allocation or possibly become unavailable; in this case it is said to be a resource with respect to that action.

NOTES

1 – For every action there is at least one participating enterprise object. Where two or more enterprise objects participate in an action, it is an interaction. When only one enterprise object participates in an action, it may be an interaction, if the object interacts with itself. [2-8.3]

2 – The specification of a role states the behaviour associated with the role, the policies applying to the role, the responsibilities associated with the role, and the relationships between roles. For example, for each role that specification includes descriptions of all actions and, for each action, identification of all the artefacts mentioned in the action and the resources used.

An actor in an action can also be an artefact with respect to that action. Likewise, an actor in an action can also be a resource with respect to that action (if it itself is used in the action).

When a resource is essential for some action, the action is constrained by the availability of that resource.

7.8.5 Process rules

In an enterprise specification, a process is an abstraction of the behaviour of some configuration of objects in which the identities of objects have been hidden as a result of the abstraction.

A process is a collection of steps taking place in a prescribed manner and leading to an objective. Each step need not be an abstraction of the behaviour of the same configuration of objects in the community.

If processes are part of a community, each step must be associated to with an actor role. A step may be associated with multiple roles.

The process specification must include specification of how it is initiated and it terminates.

The collective behaviour of a community may be represented as a set of processes. This set can be seen as a more abstract process performed by a single role performed by a c-object. Also, a step of a process can be further refined as a more detailed process.

7.9 Policy rules

7.9.1 The specification of a policy

A policy identifies the specification of a behaviour, or constraints on a behaviour, that can be changed during the lifetime of the ODP system or that can be changed to tailor a single specification to apply to a range of different ODP systems. Changes in the policies of a community during its lifetime can occur only if an enterprise specification includes behaviour that can cause such changes.

NOTES

1 – A policy is named place-holder for a piece of behaviour used to parameterise a specification in order to facilitate response to later changes in circumstances. The behaviour of systems satisfying the specification can be modified by changing the policy value, subject to constraints associated with the policy in the original specification. In these terms, a policy is an aspect of the specification that can be changed, and a policy value is the choice in force at any particular instant. Thus one might speak of a scheduling policy with a FIFO policy value.

2 – Policy may, for example, be used to configure generic components to apply them in some specific situation, or to express a pervasive decision that affects many components.

Policies may apply to a community as a whole, to enterprise objects (in all roles), to roles (for all actions named by a role or set of roles), or to an action type or set of action types named by a role or set of roles. They may also apply to the collective behaviour of a set of enterprise objects.

The specification of a policy includes:

-- the name of the policy;

-- the rules, expressed as obligations, permissions, prohibitions and authorizations;

-- the elements of the enterprise specification affected by the policy;

-- behaviour for changing the policy.

The behaviour for changing the policy may include behaviour that changes the rules of that policy and behaviour that replaces that policy with a named different policy.

NOTES

1 - The behaviour may include constraints on changing that policy

2 - Behaviour for changing the policy may be null, i.e. the policy is not changed during the lifetime of the community)

7.9.2 The specification of obligations, permissions, prohibitions and authorizations

The following sub-clauses provide a way of specifying policies:

7.9.2.1 Obligation

An obligation is defined by:

-- an authority that controls the obligation

-- an identified behaviour that is subject to that authority

-- a role or roles involved in that behaviour that are subject to the authority

-- a subset of that behaviour that is required to occur.

-- optionally, an object or objects that may fulfil the roles involved

When the obligation applies, the enterprise objects fulfilling the roles that are subject to the authority must engage in the required behaviour.

A standing obligation is an obligation that always applies.

7.9.2.2 Permission

A permission is defined by:

-- an authority that controls the permission

-- an identified behaviour that is subject to that authority

-- a role or roles involved in that behaviour that are subject to the authority

-- a subset of that behaviour that is allowed to occur

-- optionally, an object or objects that may fulfil the roles involved

When the permission applies, the enterprise objects fulfilling the roles that are subject to the authority are allowed to engage in the allowed behaviour.

NOTE – There is, however, no guarantee that the action succeeds. For example, the action may have participants in other domains in the action is prohibited.

7.9.2.3 Prohibition

A prohibition is defined by:

-- an authority that controls the prohibition

-- an identified behaviour that is subject to that authority

-- a role or roles involved in that behaviour that are subject to the authority

-- a subset of that behaviour that must not occur

When the prohibition applies, the enterprise objects fulfilling the roles that are subject to the authority must not engage in the prohibited behaviour.

NOTE – an enterprise specification may specify a behaviour by means of which the prohibited behaviour is prevented.

7.9.2.4 Authorization

An authorization is defined by:

-- an authority that controls the authorization

-- an identified behaviour that is subject to that authority

-- a role or roles involved in that behaviour that are subject to the authority

-- a subset of that behaviour that is allowed to occur

-- optionally, an object or objects that may fulfil the roles involved

When the authorization applies, the enterprise objects fulfilling the roles that are subject to the authority must not be prevented from engaging in the authorized behaviour.

Authorizations will not necessarily be effective outside the domain controlled by the authority. In federations the effect of authorizations is determined by the contract of the federation.

7.9.3 Policy violations

Some violations are the result of defective specification or implementation of behaviour. Others are caused by inconsistent assumptions of communicating parties about policies.

NOTE – These may arise, for example, in a federation where there is not full control of the interacting objects or in other situations where an action is not considered to be essential enough to be specified with policies in detail for all possible participants of an interaction.

An enterprise specification can provide mechanisms for detecting violations and for appropriate recovery or sanction mechanisms.

A way of specifying policies is as policed and enforced, or unpoliced.

If policies are specified as policed and enforced this can be specified to be by optimistic or pessimistic means.

Pessimistic enforcement is preventative and requires the specification of mechanisms to ensure that the obligated actions occur, prohibited actions do not occur, and authorized actions are not prevented. Pessimistic enforcement is specified when trust is low (i.e. when non-compliance is expected) and the damage caused by non-compliance is potentially high, and when viable preventative mechanisms can be created and/or effective sanctions can be applied after non-compliance occurs.

Optimistic enforcement is not preventative. It requires the specification of mechanisms to detect and report/correct non-compliance. Optimistic enforcement is specified when trust is high and the potential damage due to non-compliance is low, and when viable preventative mechanisms do not exist.

7.10 Accountability rules

An enterprise specification identifies those actions that involve accountability of a party

Parties can have intentions and are accountable for their actions. The concepts of subclause 6.4 are used to model an action that involves accountability of a party .

The enterprise specification identifies the actions of parties that an ODP system is prepared to participate in, respond to or record.

7.10.1 Delegation rules

An enterprise specification identifies the actions that any enterprise object that is not a party is prepared to participate in as an agent of a party. An enterprise specification describes the authority delegated to an enterprise object in terms of:

-- the parties that have delegated authority to the system;

-- the authority that each party has delegated;

-- the duration and conditions of the delegation;

-- provisions for additional delegation and withdrawal of delegation during the operation of the system.

By each such delegation, that enterprise object becomes an agent of the parties delegating, and the parties (collectively) become principal of the system. A principal is responsible for the acts of an object acting as its agent.

Insofar as provided in delegation by a party, an enterprise specification may specify further delegation, by an agent to another enterprise object.

7.10.2 Authority rules

For each authority delegated, an enterprise specification states the actions in which an agent may participate in exercising that authority. The authority delegated may be:

-- to make a commitment; this binds the principal

-- to issue a declaration; this establishes the truth of some proposition just as if the principal had made the declaration;

-- to make a prescription that establishes a rule; such a rule has the same force as if the principal had made the prescription.

-- to further delegate an authority; this causes the agent delegated to have the authority

7.10.3 Commitment rules

An enterprise specification identifies, for every commitment, the obligation created. It identifies, for every commitment made by an agent, the principal(s) obligated.

Establishing behaviour in an enterprise specification includes commitments by the objects participating in the establishing behaviour. If the establishing behaviour is implicit, it includes prescriptions that apply to the objects in the resulting liaison.

7.10.4 Declaration rules

A declaration identifies the changes that take place in the environment of an object as the result of an internal action of that object. An enterprise specification defines the conditions required for a particular declaration to be effective.

NOTE – A declaration may not be effective (cause the change in the environment of the object) until some interaction of the object such as, for example, a publication.

7.10.5 Prescription rules

An action of an enterprise object will be a prescription only when:

-- that object is a party that by its nature may establish rules,

-- that object is an agent of such a party, delegated authority to establish rules on behalf of that party,

-- the specification explicitly provides for those actions of that object that will be prescriptions, or

-- that object is, in a previous epoch, specified to establish rules.

An important special case of delegation is where the authorized action is a prescription; that is, when the delegation enables an enterprise object to make a prescription.

8 Compliance, completeness and field of application

8.1 Compliance

This document uses the term, compliance, to describe the relationship between two standards. One standard complies with another if it makes correct use of the ideas, vocabulary or framework defined there. This implies that, if a specification is compliant, directly or indirectly, with some other specifications, then the propositions which are true in those specifications are also true in a conformant implementation of the specification.

The term conformance is used for the relationship between some product or artefact and the specification from which it is produced. Conformance can be tested by inspecting the product produced to confirm the claim that its properties or behaviour are as required by the standard.

In ODP specifications, there is a need for the specifier to declare those points at which tests are to be performed and for the implementer to identify those points when offering the product for test. Large specifications are frequently organized into a specification framework populated by more detailed component specifications. The framework identifies a wide range of points at which observations can, in principle be made. These points are called reference points. The subset of reference points where tests of an implementation are required by the more detailed specifications are called the conformance points for that specification.

ODP systems are specified in terms of a number of viewpoints, and this gives rise to an accompanying requirement for consistency between the different viewpoint specifications. The key to consistency is the idea of correspondences between specifications; i.e., a statement that some terms or structures in one specification correspond to other terms and structures in a second specification.

8.2 Completeness

Specifications can be produced as a prelude to implementation, and generally change during implementation or to support system evolution. Specifications can also be produced to capture the properties of existing systems or components in order to facilitate their reuse. The references to the process of specification in this clause are intended to cover both these situations.

When a set of viewpoint specifications and correspondences is created for an ODP system, a succession of design choices is made, gradually reducing the number of conceivable implementations that would be consistent with the specification. This process is never absolutely complete, since there are always implementation choices and changes in circumstances in the environment that affect the system's behaviour, but there is some point in the design process when the specifier judges that the specification is sufficiently complete to reflect their purpose. At this point, the specification is said to have reached the viable stage. This is the stage in the specification process where it would be possible to produce some worthwhile implementation. This statement does not imply that the specification is, in any way, frozen.

The viable stage depends on the purpose of the specification, because there may be significant differences in the degree of completeness expected in, for example, an accounting policy applied to a range of independent machines or to an inter-organizational workflow. The viable stage will not be assessed to be the same for all possible applications of any particular specification notation.

8.3 Field of application

An enterprise specification includes a statement of the field of application that specifies the properties the environment must have for the specification to be applicable.

The field of application determines whether a specification is appropriate in a given situation, and must be satisfied before it makes sense to make observations of the real world and compare these with specified observable properties to test conformance to the specification.

NOTE - The provision of an accurate statement of the field of application is particularly important if reuse of the enterprise specification is expected. It allows the specifier who might incorporate the existing specification fragments to ask “is this specification for me?” before they begin to ask “what must system and its environment do?”

9 Enterprise Language Compliance

An enterprise specification compliant with this document shall use the concepts defined in clause 6 of this document and those in subclause 5.1 of ITU-T Rec. X.903 | ISO/IEC 10746-3, as well as the concepts defined in ITU-T Rec. X.902 | ISO/IEC 10746-2, subject to the rules of clause 7 of this document and those in ITU-T Rec. X.903 | ISO/IEC 10746-3 clause 5.2.

Concepts from other modelling languages may also be employed. Where such concepts are employed, the specification concerned shall include or refer to definitions of each such concept, in terms of the concepts defined in clause 6, in ITU-T Rec. X.902 | ISO/IEC 10746-2, or in clause 5.1 of ITU-T Rec. X.903 | ISO/IEC 10746-3, and explanations of the relationships between such concepts and those defined in clause 6.

10 Conformance and reference points

This standard defines the Enterprise Language, which provides a framework for a variety of notations to be used in specification. As such it creates a formal system that does not itself involve conformance, any more than, say, a programming language grammar involves conformance. However, specific notations derived from this standard will be supported by (generally automated) tools and design processes that produce and maintain enterprise specifications for systems, and the conformance of these tools and processes can be tested. This includes the generation of specifications that conform to the structural or grammatical rules of the language, and the construction of systems which, in operation, perform in a way consistent with the semantics of the language.

In general, such tools and processes manipulate not only the enterprise viewpoint specification but also manage correspondences with other viewpoint specifications, and so wider issues of conformance to complete sets of ODP specifications need to be considered.

NOTE: There are correspondences between each possible pair of viewpoint specifications, but the conformance issues involved are particularly important in this standard because the policies expressed in the enterprise specification are reflected in all the other viewpoints.

The Enterprise Language places requirements on organizational structures and business processes that cannot be observed directly, but must be deduced from the variety of interactions between the system or systems involved and their environment. In claiming conformance to an enterprise specification, the system provider must state what observable reference points in the system are conformance points, and how observations at these points can be interpreted to correspond to enterprise concepts. With this information, a tester of the system is in a position to determine by observation whether the system behaves correctly. In ODP, conformance is based on the declaration of Engineering Viewpoint reference points (in clauses 5-7 of ITU-T Rec. X.903 | ISO/IEC 10746-3), and the implementer of an Enterprise Specification must state correspondences to the engineering viewpoint in order to relate observations at the engineering reference points to enterprise concepts.

11 Consistency rules

This clause extends clause 10 of ITU-T X.903 | ISO/IEC 10746-3 by defining enterprise specification correspondences.

11.1 Viewpoint correspondences

The underlying rationale in identifying correspondences between different viewpoint specifications of the same ODP system is that there are some entities that are represented in an enterprise viewpoint specification, which are also represented in another viewpoint specification. The requirement for consistency between viewpoint specifications is driven by, and only by, the fact that what is specified in one viewpoint specification about an entity needs to be consistent with what is said about the same entity in any other viewpoint specification. This includes the consistency of that entity’s properties, structure and behaviour.

The specifications produced in different ODP viewpoints are each complete statements in their respective languages, with their own locally significant names, and so cannot be related without additional information in the form of correspondence statements. What is needed is a set of statements that make clear how constraints from different viewpoints apply to particular elements of a single system to determine its over all behaviour. The correspondence statements are statements that relate the various different viewpoint specifications, but do not form part of any one of the five basic viewpoints. The correspondences can be established in two ways:

-- by declaring correspondences between terms in two different viewpoint languages, stating how their meanings relate. This implies that the two languages are expressed in such a way that they have a common, or at least a related, set of foundation concepts and structuring rules. Such correspondences between languages necessarily imply and entail correspondences relating to all things of interest which the languages are used to model (e.g. things by objects or actions);

-- by considering the extension of terms in each language, and asserting that particular entities being modelled in the two specifications are in fact the same entity. This related the specifications by identifying which observations need to be interpretable in both specifications.

There are two kinds of standardization requirements relating to correspondences:

-- Some correspondences are required in all ODP specifications; these are called required correspondences. If the correspondence is not valid in all instances in which the concepts related occur, the specification simply is not a valid ODP specification.

-- In other cases, there is a requirement that the specifier provides a list of items in two specifications that correspond, but the content of this list is the result of a design choice; these are called required correspondence statements.

The minimum requirement for consistency in a set of specifications for an ODP system is that they should exhibit the correspondences defined in the Reference Model (part 3 clause 10), those defined in this standard, and those defined within the specification itself.

NOTES

1 – The following clauses identify the correspondences between the enterprise viewpoint and the information, computational and engineering viewpoints. Although in particular models it may be possible to establish correspondences between instances of enterprise concepts and instances of technology concepts, there are no useful generic correspondences of this nature. In particular, it should be noted that although ‘enterprise wide’ policies may exist about adoption of particular technologies, such statements are not enterprise issues as such, and should therefore appear in the technology specification for the system. Only in cases where the system has some behaviour that is related to such technology policy (for example if the system was concerned with the management of procurement of IT systems), would such policy appear in the enterprise viewpoint specification.

2 – An enterprise specification may include objects that are not part of the ODP system being specified and may include the behaviour of such objects. Where this is the case, there may be no instances of concepts in other viewpoints that correspond to these objects or their behaviour.

11.2 Enterprise and information specification correspondences

11.2.1 Concepts related by correspondences

The enterprise concepts related are:

-- community;

-- enterprise object;

-- role;

-- policy.

The information concepts related are:

-- information object;

-- dynamic schema;

-- static schema;

-- invariant schema.

11.2.2 Required correspondences

There are no required correspondences.

11.2.3 Required correspondence statements

The specifier shall provide:

-- for each enterprise object in the enterprise specification, a list of those information objects (if any) that describe part or all of its state;

-- for each role in each community in the enterprise specification, a list of those information object types (if any) that describe part or all of the state of the object fulfilling that role;

-- for each policy in the enterprise specification, a list of the invariant, static and dynamic schemata of information objects (if any) modified by the policy; an information object is included if it corresponds to the enterprise community that is subject to that policy;

-- for each action in the enterprise specification, the information objects (if any) that have a dynamic schema constraining actions in the information specification corresponding to that action;

-- for each relationship between enterprise objects, the invariant schema (if any) that represents it.

11.3 Enterprise and computational specification correspondences

11.3.1 Concepts related by correspondences

The enterprise concepts related are:

-- enterprise object;

-- role

-- enterprise interaction;

-- policy.

The computational concepts related are:

-- computational object;

-- computational behaviour;

-- computational interface;

-- operation;

-- stream

-- computational binding object.

11.3.2 Required correspondences

There are no required correspondences.

11.3.3 Required correspondence statements

The specifier shall provide:

-- for each enterprise object in the enterprise specification, that configuration of computational objects (if any) that realizes the required behaviour;

-- for each interaction in the enterprise specification, a list of those computational interfaces and operations or streams (if any) that correspond to the enterprise interaction, together with a statement of whether this correspondence applies to all occurrences of the interaction, or is qualified by a predicate;

-- for each role affected by a policy in the enterprise specification, a list of the computational object types (if any) that exhibit choices in the computational behaviour that are modified by the policy;

-- for each interaction between roles in the enterprise specification, a list of computational binding object types (if any) that are constrained by the enterprise interaction;

-- for each enterprise interaction type, a list of computational behaviour types (if any) capable of representing (i.e. acting as a carrier for) the enterprise interaction type.

11.4 Enterprise and engineering specification correspondences

11.4.1 Concepts related by correspondences

The enterprise concepts related are:

-- enterprise object;

-- role

-- behaviour;

-- interaction

-- policy

The engineering concepts related are:

-- node;

-- stub;

-- binder;

-- protocol object;

-- channel

-- interceptor.

11.4.2 Required correspondences

There are no required correspondences.

11.4.3 Required correspondence statements

The specifier shall provide:

-- for each enterprise object in the enterprise specification, the set of those engineering nodes (if any) that support some or all of its behaviour;

-- for each interaction between roles in the enterprise specification, a list of engineering channel types and stubs, binders, protocol objects or interceptors (if any) that are constrained by the enterprise interaction.

NOTES

1 – The engineering nodes may result from rules about assigning support for the behaviour of enterprise objects to nodes. These rules may capture policies from the enterprise specification.

2 – The engineering channel types and stubs, binders or protocol objects may be constrained by enterprise policies.

Index

____