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

ISO/IEC JTC1/SC7 N2187

1999/07/19

Document Type	CD Registration & CD Ballot
Title	CD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint.
Source	JTC1/SC7 Secretariat
Project	07.77
Status	
References	N2052
Action ID	FYI or ACT
Due Date	1999/10/19
Mailing Date	1999/07/19
Distribution	SC7_AG, JTC1 Sec.
Medium	Encoded Acrobat
No. of Pages	50
Note	Joint project with ITU-T (ITU-T X911).

Address reply to: ISO/IEC JTC1/SC7 Secretariat
Bell Canada - Quality Engineering & Research
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

ISO/IEC JTC1/SC7 CD 15414	
Date 1999/07/19	Reference number ISO/JTC 1/SC 7 N2187
Supersedes document **N2014**	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/JTC 1/SC 7 Committee Title **Software Engineering** Secretariat: Standards Council of Canada (SCC)	Circulated to P- and O-members, and to technical committees and organizations in liaison for: X voting by (P-members only) **1999/10/19** Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.

ISO/IEC JTC1/SC7

Title: CD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint.

Project: 07.77

Introductory note: See page ii of the document

Medium: Encoded Acrobat

No. of pages: 50

Vote on CD 15414	
Date of circulation **1999/07/19** Closing date 1999/10/19	Reference number ISO/JTC 1/SC 7 N2187

ISO/JTC 1/SC 7 Committee Title **Software Engineering** Secretariat: Standards Council of Canada (SCC)	Circulated to P-members of the committee for voting Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.

ISO/IEC JTC1/SC7

Title: CD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint.

Project: 07.77

Vote:

__ APPROVAL OF THE DRAFT AS PRESENTED

__ APPROVAL OF THE DRAFT WITH COMMENTS AS GIVEN ON THE ATTACHED

__ general:

__ technical:

__ editorial:

__ DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED

__ Acceptance of these reasons and appropriate changes in the text will change our vote to approval

__ ABSTENTION (FOR REASONS BELOW):

P-member voting:
National Body (Acronym)

Date:
YYYY-MM-DD

Submitted by:
Your Name

ITU-T X.911 ISO/IEC 15414

Date: 99-07-19

Committee Draft
Curitiba 1999 Output

8 July 1999

ISO/IEC JTC 1/SC 7

Software Engineering

Secretariat: Canada (SCC)

Doc Type:	Initial Committee Draft
Title:	Information Technology—Open Distributed Processing—Reference Model—Enterprise Language
	ISO/IEC 15414 | ITU-T Recommendation X.911
Source:	Project Editor
Project:	1.07.77
Status:	English-language initial committee draft; output from May 1999 Curitiba meeting
Action:	For National Body comment and contributions
Distribution:	SC 7
Medium:	E
Number of Pages:	53

Address reply to: joaquin@acm.org

ISO/IEC JTC1/SC7 Secretariat
Bell Canada
1050 Beaver Hall Hill 2nd floor
Montréal Québec H2Z 1S4
CANADA
Tel: +1 514 391 8286
Fax: +1 514 870 2246

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.911

(Initial Committee Draft 7/99)

COMMON TEXT DRAFT

Information Technology— Open Distributed Processing—Reference Model— Enterprise Language

ITU-T Recommendation X.911

Version 4

CONTENTS

Foreword

This is the initial committee draft of ITU | ISO/IEC Common Text for Recommendation X.911 | International Standard 15414: Information Technology—Open Distributed Processing—Reference Model—Enterprise Language. It is the draft output of the May 1999 Curitiba meeting.

The draft is in the format prescribed for ITU | ISO/IEC Common Text.

The work on this draft Recommendation | International Standard is done by ISO/IEC JTC 1/SC 7/WG 17, originally chartered by SC 21 as a project in SC 21/WG 7, and lately known as SC 33/WG 5.and SC 7/WG 3.

TEMPORARY NOTES The draft includes temporary notes, specified by the working group, for the information of National Bodies, which appear in a smaller font, indented as is this paragraph; these are not part of the text of the draft.

Editor's notes The draft includes editor's notes, which appear in a smaller font, deeply indented as is this paragraph; these are not part of the text of the draft.

Numbering Each line and page of this document is numbered, for the convenience of national bodies commenting on the document.

IMPORTANT: When referring to line and page numbers, use the numbers on the Portable Document Format (PDF) version of this document. The numbers on other versions of the document will change depending on the printer chosen while viewing the document.

Indexing The project editor requests suggestions for indexing this document.

It is the project editor's faith that a good index is an index prepared by a professional indexer.

Changes to ITU template; these must be changed back:

- Addition of style, Editors Note

- Addition of Keep lines together to Paragraph Line and Page Breaks

0 Introduction

The rapid growth of distributed processing has led to the adoption of the Reference Model of Open Distributed Processing (RM-ODP). The 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.

The Reference Model of Open Distributed 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.

0.1 RM-ODP

The RM-ODP consists of:

- 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.

- 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.

- 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.

- 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

The purpose of this Recommendation | International Standard is to:

--Refine and extend the RM-ODP enterprise language 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, so as to enable the RM-ODP to be used for specification of object-based applications architectures; 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 introduces refinements of those concepts, additional viewpoint-specific concepts, and prescriptive 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.

This Recommendation | International Standard contains, for the convenience of the reader, some text taken verbatim from clauses 5 and 10 of ITU-T Recommendation X.903 | ISO/IEC 10746-3: Part 3: Architecture. Such text is marked by a reference like this: [3-5.9], which indicates text taken from part 3, subclause 5.9 of RM-ODP. In the event of any discrepancies in these cases, the text of ITU-T Recommendation X.903 | ISO/IEC 10746-3 is authoritative.

This Recommendation | International Standard also contains some text which is a modification of text ITU-T Recommendation X.903 | ISO/IEC 10746-3: Part 3: Architecture. Such text is marked by a reference like this: [see also 3-5.9]. The modifications are authoritative with respect to the enterprise language.

This Recommendation | International Standard contains these annexes:

Annex A: Overall structure of an enterprise specification

Annex B: ODP System Rules

These annexes are not normative.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

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

1 Scope

This Recommendation | International Standard provides:

a) a language (concepts, structures, and rules) for developing, representing, and reasoning about a specification of an ODP system from the enterprise viewpoint;

b) rules which establish correspondences between the enterprise language and the other viewpoint languages to ensure the overall consistency of a specification.

This 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.

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. [3-5.0]

2 Normative references

The following ITU-T Recommendations | 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 | International 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 editions of the Recommendations | International Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The ITU-T Secretariat maintains a list of the currently valid ITU-T Recommendations.

2.1 Identical ITU-T Recommendations | International Standards

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

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

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

2.2 Paired ITU-T Recommendations | International Standards

2.3 Other International Standards

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 Recommendation 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;

-- incremental modification;

-- instance (of a type) ;

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

-- interface;

-- internal action;

-- interworking reference point;

-- invariant;

-- liaison;

-- location in space;

-- location in time;

-- object;

-- obligation;

-- ODP standards;

-- ODP system;

-- perceptual reference point;

-- permission;

-- policy;

-- postcondition;

-- precondition;

-- programmatic reference point;

-- prohibition;

-- proposition;

-- quality of service;

-- reference point;

-- refinement;

-- responding object;

-- role;

-- state (of an object) ;

-- subtype;

-- system;

-- <X> template;

-- 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 Recommendation X.902 | ISO/IEC 10746-2

-- community;

-- community template;

-- computational interface;

-- dynamic schema;

-- enterprise object;

-- <X> federation;

-- invariant schema;

-- static schema.

3.2 Definitions from ODP standards refined or extended in this standard

This Recommendation | International Standard refines or extends the definitions of the following terms originally defined in ITU-T X.902 | ISO/IEC 10746-2 (the refined or extended definitions are in clause 6):

-- policy

4 Abbreviations

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

5 Overview and motivation

TEMPORARY NOTE – The text of this clause is written from a viewpoint on “the enterprise” However, the enterprise viewpoint is a viewpoint on an ODP system and its environment [3-4.1.1.1]. An enterprise specification is a specification of an ODP system.

The working group has accepted a National Body proposal that the text be replaced with alternative text for that is written from a viewpoint on the system and its environment, and that describes an enterprise specification as a specification of an ODP system from the enterprise viewpoint.

The working group invites National Body contributions of alternative text.

The purpose of this Recommendation | International Standard is to provide a common framework for specification of the purpose, scope and policies for an ODP system, which will apply across a variety of notations and modelling methods.

Editor’s Note – The use of the term, ‘Recommendation | International Standard,’ at this point in the document contravenes clause 1 of Rules for presentation of ITU | ISO/IEC Common Text, which prescribes that “a term which is descriptive of the nature of the common text should be used when the document refers to itself” in clauses after the Scope clause. Sadly, we have no such descriptive term for this document.

The enterprise language provides the vocabulary and constructs to specify the purpose, scope and policies for an ODP system in terms that are meaningful for the stakeholders for that system. An enterprise specification describes the behaviour of the system within the social or business organization with which it interacts. Such an organization may be very large (e.g. a group of co-operating companies), more limited (e.g. a particular service inside a company), or much smaller (e.g. where a service provided by an IT system is being specified). In particular, the term “organization” is not limited to business organization.

The enterprise language defines the concepts necessary to represent the behaviour expected of an ODP system by other entities within its community, and the structuring rules for using those concepts to produce an enterprise specification.

An enterprise specification of an ODP system is an abstraction of the social or business organization of which that system forms a part, describing those aspects that are relevant to specifying what the system is expected to do in the context of the purpose, scope and policies of that organization. It describes the behaviour assumed by the users of a given ODP system. It explicitly includes those aspects of the enterprise that influence the behaviour of the ODP system – environmental constraints are captured as well as usage and management rules.

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 clients of the ODP system and the provider of that system. Both parties should be able to write, read and discuss such a specification, the clients to be sure of the expected behaviour of the system that they will get, and the provider to be clear about the behaviour to be realised by the system being provided. Thus, two types of presentation of the enterprise specification may need to be considered for the same system. One presentation may need to provide a view of the specification in terms that are understood by the clients. A second presentation may be needed to present the specification in terms that more directly relate to its realisation. Both types of presentation address enterprise considerations as they concern the system.

6 Concepts

The concepts of enterprise language defined in this Recommendation | International Standard comprise:

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

--the concepts defined in this clause.

The concepts defined in this clause include both new concepts and refinements of concepts from ITU-T X.902 | ISO/IEC 10746-2 and ITU-T X.903 | ISO/IEC 10746-3.

Editor's Note – The following classification of the enterprise language concepts is the project editor's, not the working group's. It groups related concepts, solely for the convenience of the reader.

The groupings have no normative meaning. The working group may choose to organize the concepts differently.

The names of the groups do not form a part of the draft.

The project editor invites National Body comment on the grouping and the names of the groups, including alternative organizations.

6.1 Basic concepts

6.1.1 Purpose (of a system): The practical advantage or intended effect of the system.

6.1.2 Objective (of a <Y>): Statements of preference about possible future states (not necessarily of this <Y>), which influence the choices within the behaviour of the <Y> towards the preferred future states.

TEMPORARY NOTE – There is a need to formulate a distinction between prescriptive postconditions for actions and those postconditions which include preferences about particular outcomes of an action. The former are not objectives; the latter are.

6.1.3 Scope (of a system): The behaviour named by the set of roles the system can fulfil.

6.1.4 Scoping statement: A specification of the preconditions on the use of an enterprise specification, that determine whether or not the specification can be applied in a given situation.

6.2 As yet unclassified concepts

6.2.1 Resource: An enterprise object modelling an entity which is essential to some behaviour and which requires allocation or may become unavailable because it is in use or used up.

NOTE – A consumable resource may become unavailable after some amount of use.

6.2.2 Service: A function performed by a server object (service provider) for a client object (service user).

6.2.3 Party

TEMPORARY NOTE – The working group is considering two different meaning for the term, party. To distinguish them, they have temporarily been given different names.

Editor's note – The numbers in braces (as {2}), which follow the subclause number, are shorthand for use in discussions of this subclause 6.2.4.

6.2.3.1 {1} Natural party: An object representing a natural person or any other entity considered to have some of the rights, powers and duties of a natural person.

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

6.2.3.2 {2} Contracting party (with respect to a contract): An object that agrees to that contract.

6.2.4 Owner

TERMPORARY NOTE – The working group is considering two different approaches to this concept and invites National Body comment.

Editor's note – The numbers in braces (as {2}), which follow the subclause number, are shorthand for use in discussions of this subclause 6.2.4.

6.2.4.1 {1} Owner: the default controller of an object, enabling the owner to exploit the object according to the owner's objective(s)

Editor's Notes:

1 – Would "default controlling object of an object" be better than "default controller of an object?" 'Controlling object' has a contextual definition in [2-10.3]; 'controller' is not defined.

2 – Notice that this definition may be read as implying that an object may have only one owner. This may not be intended.

6.2.4.2 {2} Owner (of a system): The party or one of several parties having the right to control the use and disposal of the system.

NOTES

1 – Commonly, this is a party paying for the specification, instantiation, construction, or current operation of the system. This party will typically grant authorisation to use the system to other parties.

2 – An owner is thus an enterprise object modelling an entity which is an actual owner of the system.

Editor's note – The meeting notes indicate that the working group has deferred any decision on this definition. The working group invites National Body comment.

6.3 Behaviour concepts

6.3.1 Behaviour (of a community):

TEMPORARY NOTE – The working group invites National Bodies to offer a definition for this concept.

6.3.2 Role (of a community): An identifier for a subset of the behaviour of a community which can be fulfilled by a single object.

6.3.3 Process: A collection of steps taking place in a prescribed manner and leading to the accomplishment of some result.

NOTES

1 – A process is distinguished from an activity, in that an activity is single-headed, while a process may have multiple heads.

2 – The activity structure concepts of ITU-T Recommendation X.902 | ISO/IEC 10746-2 subclause 13.1 may also be used to specify the structure of a process.

3 – A specification may define types of processes and process templates.

6.3.4. "Task": An action in the behavior of a community. "Tasks" are associated with roles.

TEMPORARY NOTE – The working group invites National Bodies to suggest another term for this concept.

6.3.5 Step: A "task" in a process.

6.4 Role concepts

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

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

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

6.4.3 Actor role (with respect to a community): A role in which the object filling the role is involved in at least one action of the role as an actor..

6.4.4 Artefact role (with respect to a community): A role in which the object filling the role is involved in all actions of the role only as an artefact..

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

TEMPORARY NOTE – The working group invites National Body comment on the need for these two role concepts.

Note that the concept, resource, is a third choice.

6.4.5 Core, environment, and interface objects

TEMPORARY NOTE – The working group finds that it has used certain terms in its discussions with several different meanings. The following text is offered as capturing some of these meanings.

These terms may be useful to a specifier, and might be used to reduce the size of the text at clause 7.4.3.2.

The working group invites National Body proposals as to whether any of these terms should be included, and if so, with which definitions.

Editor's note – The many level numbering scheme used in the following text is to provide convenience of reference for National Bodies commenting on this draft. Once an alternative is chosen, the numbering of concepts will be three level only, as in the rest of clause 6.

The numbers in braces (as {2.2}), which follow the subclause number, are shorthand for use in discussions of this subclause 6.4.5.

6.4.5.1 Objects defined in terms of roles

6.4.5.1.1 {1} Core object: An object fulfilling a core role.

6.4.5.1.2 {2} Environment object: An object fulfilling an environment role.

6.4.5.1.3 {3} Interface object: An object fulfilling an interface role.

6.4.5.2 Roles defined:

6.4.5.2.1 First alternative

Editor's note – The community referenced by 'that community' in the following definition, 6.4.5.2.1.1, is clear if this definition is placed directly after the first paragraph of the text which preceded the definition in the working document in which these definitions appear. Clearly, the definition will need to be rewritten if it is to stand alone in clause 6. That text in the working document is:

"Any enterprise object may be decomposed into a configuration of objects that may be represented as a community or as some of the objects in a community that includes other objects."

6.4.5.2.1.1 {1.1} Core role: A role to be fulfilled by an object in that community.

6.4.5.2.1.2 {1.2} Environment role (with respect to a community): A role to be fulfilled by an object outside the community that interacts with objects of the community.

6.4.5.2.1.3 {1.3} Interface role: A role to be fulfilled by an object that interacts with objects outside the community.

6.4.5.2.2 Second alternative:

6.4.5.2.2.1 {2.1} Core role (with respect to a community): A role to be fulfilled by an object in that community that is a composite object which, in some other system description, is represented as a community.

6.4.5.2.2.2 {2.2} Environment role (with respect to a community): A role to be fulfilled by an object outside the community that interacts with objects of the community.

6.4.5.2.2.3 {2.3} Interface role: A role to be fulfilled by an object that interacts with objects outside the community.

6.4.5.2.3 Third alternative:

6.4.5.2.3.1 {3.1} Core role (in a community): A role in a community that is central to the objective of the community.

6.4.5.2.3.2 {3.2} Environment role (with respect to a community): A role to be fulfilled by an object that interacts with objects of the community but is not a part of the community.

6.4.5.2.4 Fourth alternative:

6.4.5.2.4.1 {4.1} Core role (in a community): A role in a community that is central to the objective of the community.

6.4.5.2.4.2 {4.2} Environment role (with respect to a community): A role to be fulfilled by an object that interacts with objects of the community but which role is not central to the objective of the community.

6.4.5.2.5 Fifth alternative:

6.4.5.2.5.1 {5.1} Core role (in a community): A role in a community to be fulfilled by an object that has agreed to the community contract.

6.4.5.2.5.2 {5.2} Environment role (with respect to a community): A role to be fulfilled by an object that has not agreed to the community contract.

TEMPORARY NOTE – The working group considers that it is desirable to describe these matters first in terms of actions, and then, if required, to speak of roles and interfaces. When the description is done first in this way, in terms of interactions, a description in terms of roles or interfaces will be trivial.

The working group invites National Body proposals.

6.5 Policy concepts

6.5.1 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.

NOTE – Not every policy is a constraint. Some policies represent an empowerment. This defininition refines [2-11.2.7].

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

NOTE – Unlike a permission, an authorisation is an enpowerment

6.5.3 Violation: An action contrary to a rule.

TEMPORARY NOTE – The working group invites National Body comment on structuring rules using this concept.

The project editor invites National Body comment whether this concept is redundant, and is covered adequately by the concept, failure, of 2-13.5.1.

6.6 Delegation concepts

6.6.1 Delegate: To give (authority, a function, etc.) to another.

TEMPORARY NOTE – The working group invites National Body comment on structuring rules using this concept.

6.6.2 Agent: An object which has been delegated (authority, a function, etc.) by and acts for another (in exercising the authorisation, performing the function, etc.).

NOTES -

1) An agent may be a party.

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.

TEMPORARY NOTE – The working group invites National Body comment on structuring rules using this concept.

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

6.7 Force concepts

TEMPORARY NOTE – The working group invites National Body input on structuring rules using these concepts.

6.7.1 Act: An action in which the initiating object is a party or agent and which action is a model of something done by an entity modelled by a party or its agent.

6.7.2 Commitment: An act by which an object is bound by a rule or contract. A commitment creates an obligation to comply with the rule or perform the contract.

6.7.3 Declaration: An act that constitutes a state in the universe of discourse modelled by the environment of the system.

NOTE – The essence of a declaration is that, by virtue of the act of declaration itself, it causes a state of affairs to come into existence in the universe of discourse.

6.7.4 Delegation: An act which delegates (authority, a function, etc.)

Editor’s note – This concept and definition may not be needed. It is included in this draft for grammatical parallelism with the other speech act definitions. On the other hand, it may be desirable to restrict its meaning to: An act… Otherwise it will have the ordinary English meaning that follows from the earlier definition of ‘delegate.’

6.7.5 Description: An action which communicates information about the representation within the system of a state in the universe of discourse modelled by the environment of the system.

Editor’s note – Is this an act? The answer seems to be, not usually, though it might be in some cases.

6.7.6 Evaluation: An act which assigns a value to something.

NOTE – For example, the act by which a system assigns a relative status to some thing, according to an estimation by the system of its worth, usefulness, or importance.

6.7.7 Instruction: An act which is intended to cause a person or machine to do something.

6.7.8 Prescription: An act which establishes a rule.

7 Structuring Rules

An enterprise specification for an ODP system is a model of that system and its environment that focuses on the scope and purpose of that system and the policies that apply to it in the context of the organisation in which it operates. The model is specified in terms of a structure of communities. A community is a configuration of enterprise objects representing a group of entities in the organisation (e.g. human beings, IT systems, resources of various kinds and groupings of these) that are subject to some implicit or explicit agreement governing their behaviour in the organisation.

Note: The term "organisation" is not limited to business organisation.

This clause defines how the concepts identified in clause 3 or defined in clause 6 of this Recommendation | International Standard are used in an enterprise specification.

An enterprise specification defines, and the enterprise language is able to express, the purpose, scope and policies of an ODP system in terms of each of the following items:

-- roles played by the system;

-- activities undertaken by the system;

-- processes in which the system participates;

-- policy statements about the system, including those relating to environment contracts.

7.1 Overall structure of an enterprise specification

In an enterprise specification, an ODP system and the environment in which it operates are represented as a community. At some level of description the ODP system is represented as an enterprise object in that community. The objectives and scope of the ODP system are defined in terms of the roles it fulfils within that community, and policy statements about those roles.

An enterprise specification includes the specifications of:

-- that community,

-- any other communities of which the system or its parts are members, and

-- other communities of which objects in the environment of the system are members.

NOTES –

1 - Types of communities or community templates may be used in the specification of these communities.

2 - Types of communities may be related by refinement.

3 - An enterprise specification will define communities that include objects in the environment of the system, but not the ODP system or its parts, when this is necessary or will clarify the specification.

7.2 [was 7.3] Community rules

7.2.1 Specification of a community

A community is defined in terms of each of the following elements:

-- the enterprise objects comprising the community;

-- the roles fulfilled by each of those objects;

-- processes which take place in the community;

-- policies governing interactions between objects fulfilling roles;

-- policies governing the creation, use and deletion of resources by objects fulfilling roles;

-- policies governing the configuration of objects and assignment of roles to objects;

-- policies relating to environment contracts governing objects in the community.

An enterprise object may be part of a community because:

-- the specification of that community provides that the community includes the object;

-- the object is made a part of the community at the time the community is created; or

-- the object becomes a part of the community as a result of the operation of a dynamic mechanism which changes the configuration of the community.

Interactions between enterprise objects fulfilling appropriate roles within different communities can be considered as interactions between those communities.

7.2.2 Establishment of a community

A community is established by a contract, an agreement between *parties* (enterprise objects). The objective of the community will be a subset of the objectives of all of the *parties* of that community contract. The *parties* to the community contract are (collectively) the *owner* of the community upon instantiation.

NOTES:

1 – There is no requirement that the *parties* fill other roles in the community. Indeed, the lifetime of a community can exceed the lifetime of the *parties* to the community contract.

2 – It is the modeller's choice whether the *owner* of a community is of relevance and, if so, whether the *owner* should be modelled as a community whose roles are filled by the *parties* to the community contract.

Editor's note – The term, party, is used in this subclause with a meaning of subclause 6.2.3.2 The working group has not resolved the meaning of the term, owner. The terms 'owner' and 'party' are in italics as a reminder of the differing meanings of these terms.

7.2.3 Relationships between communities

An enterprise specification may specify several communities. A community need not be specified in isolation; rather, it may be considered in the context of some other community or communities to which it is related. These communities may be related in various ways, including relationships when:

-- the same object fulfils roles in each of the communities;

-- the enterprise specification prescribes that the object fulfilling a role in one community be part of other communities, perhaps fulfilling a certain role in each of those communities;

-- the communities interact.

NOTES:

1 – When objects in different communities interact, this need not be considered as an interaction between those communities, when the interaction is not part of the roles the objects fulfil in those communities.

2 – When the same object fulfils roles in several communities, the communities need not be considered to be related, if the actions of the object in each of the communities are unrelated to its actions in the other community.

An enterprise object at one level of abstraction may be represented as a composition of objects at another level of abstraction. Communities may thereby be related to each other in further ways:

-- a community is represented as a composition of objects in other communities;

-- a composite object is part of a community and (some of) its component objects are parts of other communities.

NOTES:

1 – A specification may prescribe, as a part of the specification of a community, that a certain object in the community is a composite object represented, at a different level of abstraction, as another community. Or it may happen that an object that is specified to be, or becomes a part of the first community happens to have been specified elsewhere as a composition of the objects in some other, otherwise unrelated community.

2 – If two communities share a fragment of specification (such as a role template or policy statement) the communities are not necessarily related.

Each of these types of relationships is transitive (that is, if community A is related to community B by one of these types of relationships, and community B is related to community C by the same type of relationship , then community A is related to community C by the same type of relationship).

NOTE – The first case is where it is necessary to consider the community of interest (perhaps one in which the ODP system appears as a single object) in the context of some larger (outer) community, in which it is modelled as an enterprise object fulfilling a role. For example, if a community interacts with its environment (i.e. with objects that are not members of that community) then it does so as a part of (i.e. as an enterprise object subject to the policies of) a community of greater scope. In such a case, the community being defined is an expression as a composite object of an enterprise object in the community of greater scope.

There may, also, be cases where it is necessary to specify peer-to-peer interaction between communities. For interaction between the communities to be meaningful, there must be some element of shared objective, which itself implies a higher level of community of which both communities will be members, and a common set of policies will apply.

NOTE – Some examples may help to clarify this.

A company may be defined as operating within a particular legal system, for example under English Law. This is equivalent to saying that there exists a community whose members are those subject to English Law. The laws and decision procedures can then be codified as the behaviour of this outer, English Legal community. By being defined within this context, the community representing the company inherits from the outer community a corresponding set of obligations on its members, representing the requirement that they operate in accordance with the law.

At a rather narrower level, broad communities may be defined to represent aspects of commercial activity, such as buying and selling, or even a specific framework for ownership of resources. A specific commercial undertaking or organisation may be modelled in terms of a community specification identifying its constituent members, the roles they fulfil and the internal procedures under which they operate. Some of these roles may be required to operate on behalf of this organisation as, for example, buyers for the organisation, and this part of their behaviour can be expressed by requiring that they play a buyer role within the definition of the (outer) commercial community. Making this association opens up the ability to define "suppliers" to the organisation as the set of things having seller roles in the (outer) commercial community definition. The commercial obligations on these roles then follow directly without re-specification.

Another example might be of an outer community that requires an auditor role to police the behaviour of core communities. Depending on the size and complexity of the organisation being described, this may lead to the definition of a further core community defining the procedures of an audit office to fulfil this role.

7.3 [was 7.4] Objective rules

TEMPORARY NOTE - The working group invites National Bodies to comment on this clause.

Every community has exactly one objective, which is defined in its contract. Objectives can be refined into a collection of sub-objectives.

Objectives of a community can be realised by a collection of roles, processes, and policies. A core role can have an objective, which is to be met by the object filling that role. Execution of processes meets objectives by performing that part of the community behaviour that can be prescribed. Policies describe the parts of the community behaviour which are not yet prescribed, but which influence decisions within behaviour on a case-by-case basis in order to steer the behaviour towards an objective. Typically roles, processes and policy will be designed to meet different each sub-objective. A step in a process can be a policy-influenced decision, which may in turn result in creation (or termination) of processes; hence processes and policies can be intertwined.

Objects can have objectives, if the entity being modelled is capable of having objectives. Objects filling roles must take on the objectives of those roles.

Where a community is filling a role in a community of wider scope, the objective of the first community shall be consistent with the objective of its role in the second community.

7.4 [was 7.5] Behaviour rules

Editor's note - The editing instructions from the Curitiba meeting included text for two different subclauses numbered 7.5.1 (which number would now be 7.4.1). Accordingly, these are included as subclauses 7.4.1 and 7.4.2, and the remaining subclasues of subclause 7.4 are renumbered.

7.4.1 Objects and actions

An object specified in an enterprise specification is involved in at least one action that is a part of the behaviour of a community. Objects can be involved in actions in one or both of two ways:

-- An object can participate in an action.

-- An object can be referenced in an action.

For every action there is at least one participating object. Where two or more objects participate in an action, it is an interaction.

Editor's note - When only one objects participates in an action, it may be an interaction, if the object interacts with itself.

An object that participates in an action is said to be an "actor" with respect to that action.

An object that is referenced in an action is said to be an "artefact" with respect to that action.

An object that is used in an action is said to be a "resource" with respect to that action.

In the special case where an object is referenced in an action in which it also participates (e.g. some object reporting its state) it is both an actor and an artefact with respect to that action.

Resources can be used in an action, and the occurrence of the action is constrained by the availability of those resources. There can be zero or more resources associated with each action.

Where, in some community, a role is involved in actions only as an artefact, then it is an artefact role in that community. If a role is involved in any action as an actor, then it is an actor role in that community.

NOTE - Therefore, roles in a community can be partitioned into actor roles and artefact roles with respect to that community.

7.4.2 Roles and processes

TEMPORARY NOTE – The following text was drafted during the Curitiba meeting as a proposal. The working group invites National Body comment.

TEMPORARY NOTE – This text uses "participate in an action" rather than "perform an action." It will be necessary to finally resolve the form of expression of this concept.

The behaviour of a community defines what the community should be observed to do and consists of a number of expected actions, called "tasks." The behaviour defines the possible ordering of the "tasks".

NOTE - There are many specification styles for expressing the ordering of "tasks." The modelling language chosen for expressing an enterprise specification may impose certain styles.

Some parts of object behaviour may not be relevant, and must be hidden by abstraction. Two approaches for expressing the behaviour of a community are identified in this Recommendation | International Standard:

-- Role-based approach: a collection of abstractions of the community behaviour in which each abstraction includes only those "tasks" which are related to a particular action within the community. Each abstraction is labelled with a role name. The emphasis is on which objects participate in the behaviour.

-- Process-based approach: a collection of abstractions of the community behaviour in which each abstraction includes only those "tasks" which are related to achieving some particular result/purpose/sub-objective within the community. Each abstraction is labelled with a process name. The emphasis is on what the behaviour achieves.

TEMPORARY NOTE – The working group notes that collective behaviour needs to be considered here

Role behaviour decomposes the behaviour of the community into roles, that can each be performed by an object in the community. The object that performs the role behaviour is said to fulfil that role within the community or is said to be assigned to that role within the community. The behaviour of a role will be a subset of the behaviour of the object that fulfils that role. An object can fulfil many roles within the same community (or indeed, many roles within different communities).

Each "task" will be part of at least one role behaviour, but can be part of many role behaviours (e.g. when the "task" involves an interaction).

Editor's note - Or involves collective behaviour.

Process behaviour decomposes the behaviour of the community into processes. Each step need not be performed by the same object in the community. When a "task" occurs in a process, it is known as a step reflecting the sequencing of the "tasks" in the process.

A "task" can be part of zero or more processes. That is, while a "task" is performed by a role, it may or may not be part of a process (i.e. it may or may not be a step).

The choice of which approach to use will depend on the modelling method used and the aim of modelling; it is possible to use a combination of both. However, if the process-based approach is used:

a) It is necessary to specify:

1) the association of each step with the roles that perform the step

2) the association of each step with the roles which the step references

3) the assignment of the objects to the roles.

b) The "tasks" which are not part of a process (i.e. are not steps) are specified for each role, together with any ordering constraints applicable to those "tasks."

TEMPORARY NOTE – The working group invites proposals from National Bodies on selecting a single word for "scope/purpose/objective" and "result/purpose/sub-objective" in this clause.

The choice of which approach to use will depend on the modelling method used and the aim of modelling. As a minimum the roles of the objects in the community shall be specified. If the process-based approach is used as well, it shall be in accordance with the rules for the specification of processes given in 7.4.4.

7.4.3 Role rules

In the specification of a community, each role stands as a placeholder for some object.

An object may fulfil several roles in one community, and may fulfil roles in several communities. Different objects may fulfil a role in a community at different times, or by several objects at the same time.

When a community template is instantiated, one or more objects is associated with each role. The constraints of the behaviour named by the role become constraints on the object(s) fulfilling the role.

Editor's note – Couldn't it be the case that no object is be associated with some role?

During the life of a community roles may be added or removed.

Editor's note – Would it be better if a community specification prescribed the number of objects in each role, that number could change, and could be zero?

The policies of a community apply to the behaviour of the objects in the community. Policies of the community determine the assignment of roles to enterprise objects.

NOTE – The emphasis in defining a community is on the specification of a state of affairs (the existence of the community) in terms of the behaviour expected of any enterprise objects that are to be involved in it. The focus is on the shared behaviour expected, and the community comes into existence when some initial set of enterprise objects take up this behaviour. The behaviour is seen as continuing throughout the lifetime of the community, although the set of enterprise objects involved may change during this period, and the details of the behaviour may change as a result.

The concept of role de-couples the expected behaviour from the identities of particular enterprise objects. A role is a placeholder (a formal parameter) providing an identifier for some part of the community behaviour. Associated with a role is a set of constraints on the behaviour expected of any enterprise object that is to fulfil the role; these are requirements on the candidate enterprise object type.

Role is a specification concept; it is generally used as part of a type specification to link pieces of independent specification together in a consistent and type-safe way. Each role stands as a placeholder for some enterprise object, and therefore has a type that is closely related to an enterprise object type, but it is not, as such, an instance of that type. When the community type specification containing the role is instantiated to create a composite object (a community), one or more specific enterprise objects are associated with (fulfil) the role, thus constraining relevant aspects of the behaviour of those objects.

The roles in a community may vary during its lifetime, since its behaviour may evolve. Roles may be created or destroyed, so that 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.

Editor's note – S T-B asks: How does one destroy a name?

A focus in describing a community is therefore on the consequences of the community being in existence, as well as on the interaction of its members. What emerges from the community behaviour is a series of constraints on the behaviour of those members – a set of community policies.

In general, one enterprise object may fulfil many roles, in any number of communities, although in special circumstances there may be constraints preventing one enterprise object from fulfilling conflicting roles. The enterprise object that fulfils a role does not have to be of precisely the type specified for the role (it may have some additional capabilities), although there will generally be eligibility rules specific to the specification context. These may be type matching rules (e.g. of the kind defined in ITU-T Rec. X.903 | ISO/IEC 10746-3, clause 5, for the computational language, based on signatures), but they will, in general, involve behaviour; in particular, they often involve obligations to restrict the behaviour of the enterprise object fulfilling the role by not initiating some of the behaviour the enterprise object is, in fact, capable of.

There are two pairs of sub-classifications of roles that are of interest to enterprise specifications:

-- actor roles and artefact roles.

-- environment roles and core roles

The choice of how each type of role is used in any model will depend on the purpose(s) of the model.

7.4.3.1 Community structure

Objects of a community may interact with objects outside of that community. In this case, an enterprise specification identifies the roles in that community that include interactions with objects outside of that community. A description of a system in an enterprise specification may identify roles to be fulfilled by those objects outside that community that interact with objects of the community.

In a second, different description of the system, in that same specification, that community may be considered as a single composite object. In this case, an enterprise specification specifies the interactions with objects outside of that community as interfaces of the composite object.

The following structuring rules prescribe how the interactions in the community roles in the first system description are mapped to the interfaces of the composite object in the second system description:

TEMPORARY NOTE - Here we are missing the specific structuring rules that answer questions raised by the Australian delegation. The working group invites National Bodies to propose structuring rules.

This composite object may be a member of a larger community. In this case:

-- the composite object fulfils roles in the larger community; and

-- the other objects of the larger community interact with the composite object through its interfaces.

The following structuring rules prescribe how (in the second system description) the actions in the interfaces of the composite object and the internal actions of the composite object are mapped to the roles of the larger community:

TEMPORARY NOTE - Here we are missing other specific structuring rules that answer questions raised by the Australian delegation (or no rules are necessary). The working group invites National Bodies to comment or to propose structuring rules.

The following structuring rules prescribe how the collection of actions in the other roles of the larger community in the second system description are mapped to the collection of roles to be fulfilled by those objects outside the community in the first system description:

TEMPORARY NOTE - Here we are missing specific structuring rules (or no rules are necessary). The working group invites National Bodies to comment or to propose structuring rules.

These rules, in combination with the preceding rules, provide a mapping of the roles of the first community to roles of the larger community.

The same composite object may also be a member of other communities.

Likewise, any enterprise object may be decomposed into a configuration of objects that may be represented as a community or as some of the objects in a community that includes other objects.

TEMPORARY NOTE – The preceding text is careful to state the rules without using any defined terms for the different communities. Instead, it mentions a particular community ("that community"), objects outside that community, and a possible larger community. Terms that might have been used are defined in subclause 6.4.5.

NOTE - Roles may be unique within a community (such as the chairman of a committee, seen as a community) or may be sets, generally with cardinality constraints (such as the membership of the committee). Depending on the constraints given, some of the roles will need to be associated with the objects that are to fulfil them at the time when the community is created. Others may be fulfilled by specific objects during the lifetime of the community.

For example, in modelling some commercial activity the roles of salespersons, sales-managers and customers may be identified. It is quite possible to describe the community as having these three role types, noting that the lifetime of an object in a customer role is likely (in many cases) to be very much shorter than the time during which particular objects fulfil the other two roles. However, the identity of the particular object assigned to the customer role may not be of great importance; it is sufficient to know that there is a set of objects, each available to fill the role at various times in the life of the community.

In such circumstances, it seems artificial to treat the three roles as equal, speaking of the community as having three role-types, all of which are, equally, members of the community. It is more useful to give some prominence to those roles which have some permanence and which involve obligations and commitments to the community.

Editor's note – Here the editing instructions say "picture into text," meaning that what is shown in Figure is to be presented as text. Your project editor has failed in the effort to do that.

7.4.3.2 Interface Roles

It is sometimes necessary to consider a community as providing one or more services. This can be modelled by considering each service as an interface of the CEO. Then, in specifying the core community it will be necessary to establish the mapping between the interfaces (services) of the CEO representing the core community and the enterprise objects in the core community that fulfil roles directly associated with providing the service(s). These are interface roles. This mapping emphasises the fact that interactions must be modelled as being between objects of the core community and objects who belong to the outer community (where the CEO is an object), rather than as being between the CEO and objects in the outer community.

Thus, in a core community an interface role is a core role which represents an interface of the corresponding CEO. The CEO fulfils the interface roles of its corresponding core community. In an enterprise specification it is necessary to establish a link between each interface role (of the core community) and objects that make up the core community. An object fulfilling an interface role is responsible for provision of some aspect of the CEO's services.

7.4.3.3 Specifying roles

Roles should be specified in such a way that the behaviour associated with them, the constraints on that behaviour, the responsibilities associated with the role, and the relationships between roles are unambiguous. This implies that, for every actor role there should be complete descriptions of all actions that it performs, and for every action, identification of all the artefact roles mentioned.

7.4.4 Process rules

7.4.4.1 Specifying processes

Processes are modelled as sets of steps, in which each step:

-- is triggered by one or more event or state change of an enterprise object, or the completion of one or more previous steps in the process;

-- results in some one or more defined state (result) or a trigger for one or more subsequent actions in the same process.

An action in a process may itself be modelled as a (sub-)process.

TEMPORARY NOTE – The preceeding and following texts are alternatives being considered by the working group. The working group invites National Bodies to make proposals.

A process is modelled as a directed acyclic graph of steps, with a set of constraints on when they may occur, where the occurrence of each action in the graph is made possible by the occurrence of all immediately preceding actions (i.e. all actions leading to that action).

Processes are specified as sets of actions, in which action:

-- is made possible by events or state changes due to the occurrence of preceding actions;

-- results in state changes, some of which make possible the occurrence of subsequent actions in the same process.

NOTES

1 – The use of 'acyclic' indicates that the trace, or history, of actions does not contain cycles of cause and effect. This does not prevent the use of notations with a concept of iteration; such looping concepts generate a sequence of distinct action occurrences.

1 – In an enterprise specification, a process is an abstraction of the behaviour of some community in which the identities of the objects fulfilling roles in the community have been hidden as a result of the abstraction.

Editor's note – Do we want "some configuration of objects," as more general than "community?"

7.4.4.2 Relationships between process and other enterprise language concepts

Each action identified in a process should be associated with some actor role. Different actions in a process can be associated with different actor roles or different enterprise objects. One of the purposes of modelling processes is to capture how the behaviour identified by roles can be logically composed to achieve the objectives of a community.

The results of actions in a process can represent the flow of information round a community, and as such may have identified associations with artefacts and with the interactions between actor roles.

7.5 [was 7.6] Policy rules

TEMPORARY NOTE – This material may not be normative. If it is, it needs to be rewritten. The working group invites National Bodies to propose rewrites of normative policy structuring rules for clause 7, or informative material for an annex.

The policies of a community prescribe behaviour or the membership of a community. One form of behaviour that may be prescribe is the action of making policy, and so policies may also be thought of as prescribing other policies of a community.

Policies can be implicit or explicit. Implicit policies are those that are expressed completely within the specification of some role, enterprise object or action. Explicit policies are those are specified in their own right. This sub-clause is about explicit policies.

Some of the ways in which creation of a community changes the policies that apply to its members are:

-- the enterprise objects fulfilling actor roles agree, by participation in the behaviour which creates the community or introduces them to it, to be subject to policies; their behaviour is then more restricted than it was before;

-- an enterprise object fulfilling one of the actor roles may, by virtue of its role in some outer community, invest the community with responsibility to impose policies on other enterprise objects fulfilling roles in some inner community; this implies that the acceptance of such policies from proper authority was already prescribed in the outer community;

-- enterprise objects fulfilling actor roles may undertake obligations not to do things unless specifically allowed by the rules of the community; this gives rise to a structure of permissions, administrated by some role in the community.

7.5.1 Scope of policies

Policies have varying scope. A policy may apply across a community (e.g. all members of the community shall be paid-up subscribers), to a role (e.g. any object fulfilling actor role <X> may not also be a member of community <Y>) or to a single action type (e.g. invoices shall not be paid until at least 30 days after receipt).

Within a community, policies are used to express constraints on the behaviour of objects fulfilling actor roles. It is here that the environment in which the community is specified becomes particularly important. No community can operate in isolation from its context; it cannot arbitrarily place or relax policies, but is itself constrained by what it inherits from the environment in which it is created and the obligations taken or responsibilities devolved by its members.

TEMPORARY NOTE - Similar material appears in clause 8. The working group invites National Bodies to comment.

NOTE – The scope for establishing a policy framework is limited to the enterprise objects that fill roles in the community and anything over which they have effective control (this control might arise through participation in other communities, or through ownership).

Policy can reference enterprise objects that do not meet the criteria above, but it must be made clear which enterprise objects are actually obligated. There is a significant difference between "The members of this bushwalking club shall not cut down the trees in the forest" and "Nobody shall cut down the trees in the forest", where the trees in the forest are not under the control of the bushwalking club. The first one obligates the members of the community (this is valid); the second one attempts to obligate people outside the community (and is not valid). A more appropriate/realistic expression of the latter obligation would be "Members of this community will behave in such a was as to prevent the cutting down of the trees in the forest"; this puts the obligation back onto the enterprise objects who can be constrained by the policies of this community.

Of course, if an outer community gave the bushwalking club the right to make policy about the trees in the forest, then members of the outer community would be constrained by the policies of the bushwalking club, through common membership of the greater, outer community.

In general, this concept of an outer community needs to be brought into play whenever the problem of conflict between the policies of autonomous communities arises. In this way, policies can be defined with regard to some form of community, albeit sometimes a rather loose one. Where there is no acknowledged outer community, it may not be possible to resolve conflicts in a civilised way, and disputes may be ongoing, or resolved by possession or force.

In case of conflict between policies of competent communities, conflict resolution may be accomplished by policies of a outer community that has a conflict resolution authority, or by a set of existing acknowledge policies of an unspecified outer community, or by creation and enforcement of such acknowledged policies by some or all component communities.

7.5.2 Ownership

The owner of an owned object is its default policy maker / controller for the owned object. That is, the owner makes policy for the owned object, except for those aspects of the owned object for which the owner is prohibited from controlling or for which the owner has relinquished its control. Each object has at most one owner (although the owner can be refined or delegated to a set of cooperating objects). An object can be owned by itself. An object can own any number of objects. When an object is instantiated, its ownership is specified. Ownership can be transferred or delegated (temporarily or permanently) or can be relinquished (permanently).

NOTE - The above describes the most unconstrained circumstances. In many cases, there will be background policies constraining, e.g. what can/must be owned, what can/must be an owner, what can/must own what, under what circumstances ownership can be transferred/delegated/relinquished.

7.5.3 Permissions, prohibitions and obligations

Policies that constrain behaviour are limiting statements about the choices that an enterprise object has in fulfilling an actor role. In other words, if some behaviour in a role is constrained by a policy, the enterprise object filling that role has no choice about obeying that constraint.

Policies are generally (but not always) expressed in terms of permissions, prohibitions and obligations. Such constraints may apply to enterprise objects (in all roles), 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. Policies may also apply to collective behaviour of a set of objects.

TEMPORARY NOTE – The following subclauses, 7.5.3.1 thru 7.5.3.3, may be overly prescriptive. The working group invites National Bodies to propose text.

Keep in mind that rules may mention more than one object or role.

Consider mention of collective behaviour.

7.5.3.1 Permission

ITU-T X.902 | ISO/IEC 10746-2 defines permission as "a prescription that a particular behaviour is allowed to occur. A permission is equivalent to there being no obligation for the behaviour not to occur."

A permission is defined by:

-- an action

-- a role involved in that action

-- a predicate on behaviour

-- an authority which grants the permission

If an enterprise object has this permission, then, when the predicate is true, the enterprise object can take part in the action when filling the role, by order of the authority.

There are two ways of looking at the concept of permission. The text in ITU-T X.902 | ISO/IEC 10746-2 follows standard deontic logic in describing the way the world is, in terms of what actions may occur; this corresponds to the idea of "having permission". However, there is another common usage, associated with "granting permission", in which there is an implicit or explicit agency, and there are consequential obligations on the authority as a result of granting a permission. The authority should not normally simultaneously grant a permission and prevent the permitted action from taking place. Thus, the second statement in the definition from ITU-T X.902 | ISO/IEC 10746-2 above is weaker than the first, because it does not acknowledge that permission can be "empowering". If X is permitted to do Y, then X has the luxury of choosing whether or not to do Y, but the authority has no such choice; it is obligated to allow X to do Y.

TEMPORARY NOTE – The concepts of granting and having permission have been removed from clause 6. The preceeding paragraph needs to be rewritten. The working group invites National Bodies National Bodies to propose text.

7.5.3.2 Prohibition

ITU-T X.902 | ISO/IEC 10746-2 defines prohibition as "a prescription that particular behaviour must not occur. A prohibition is equivalent to there being an obligation for the behaviour not to occur."

Like a permission, a prohibition is defined by:

-- an action

-- a role involved in that action

--a predicate on decisions influenced by policy

-- an authority which imposes the prohibition

If an enterprise object has this prohibition, then, when the predicate is true, the enterprise object cannot take part in the action when filling the role, by order of the authority.

7.5.3.3 Obligation

An obligation is a prescription that particular behaviour is required. An obligation may be prescribed, and then fulfilled by the occurrence of the prescribed behaviour.

Editor's Note - This change is made in accordance with editing instructions. It changes the definition of obligation in Part 2. Accordingly, a refined definition of obligation is needed in clause 6.

The obligation to obey the policy-making of the community is the basis on which communities grant permissions and impose prohibitions. Thus, a community may create a context in which the acceptance of permissions, prohibitions or further obligations is itself obligatory, based on prior agreement by the members of the community.

Editor's Note - The editing instructions say to make explicit reference to composition in the preceding paragraph. I have failed to do that, not seeing how.

Obligations can be expressed as:

1) enabling (triggering) conditions, which makes the obligation “active”. This may be expressed either as:

   a) a predicate that holds while the obligation is “active”, e.g. “when it is dark, you will watch my house”

   b) a pair of activating and deactivating conditions which toggle the obligations into active and inactive modes, e.g. “when the sun sets, you must start watching my house and continue to do so until the sun rises again”.

2) satisfaction condition, which signifies the obligation has been satisfied, e.g. “you must pay me $10”

3)- violation condition, which signifies the obligation is unachievable, e.g. a deadline has passed

All of the above might be expressed in terms of predicates on states, or the occurrence of some behaviour.

TEMPORARY NOTE – The concept of trigger conditions, while interesting and useful, raises several issues:

-- Many obligations, being prescribed in a specification, will always apply. It seems awkward to be required to specify a trigger condition in such cases.

-- It is not clear why there are not trigger conditions on permissions.

-- Given that a prohibition is a kind of obligation, it seems to follow that there will be trigger conditions on prohibitions.

Likewise, with respect to the introduction of satisfaction and violations conditions here, but not on prohibitions.

In any case, precisely what form is used to specify an obligation will depend on the chosen specification language. The standard should not specify a particular specification language for rules.

Standing obligations can never be satisfied, so these must be defined by a violation condition.

TEMPORARY NOTE – The working group invites National Bodies to propose a definition for standing obligation.

TEMPORARY NOTE – If the specification forms for rules are kept, one should be added for authorisation. The working group invites National Bodies to comment or to propose text.

7.5.4 Nesting of policy frameworks

TEMPORARY NOTE – The working group invites National Bodies to offer an explicit description of a policy framework and rules for nesting such frameworks at an abstraction level suitable for this standard. Until then, the text formerly in this subclause has been moved to a text parking lot at the end of this draft.

7.5.5 Enforcing policy

Depending on the contract for the community, policies may be policed and enforced or unpoliced. If policies are policed and enforced this can be by optimistic or pessimistic means.

Pessimistic enforcement is essentially preventative and involves on-going checking. Mechanisms are devised to ensure that the right things are done and the wrong things are not. Real world examples include locking a car to prevent access except by those with keys, checking of a security pass on entry to a building, etc. Passwords and access control lists are used in computer systems. Note that all of these examples are primarily concerned with preventing the prohibited actions. It is more difficult to devise mechanisms to force required things to happen. However, there are some examples, e.g. an alarm clock can be set to ensure that the person wakes up on time. Generally pessimistic enforcement of obligations tends to take the form of nagging, "You still haven't done X", i.e. constant reminders of the obligation.

Pessimistic enforcement tends to be used:

-- when trust is low, i.e. when the community has the belief (rightly or wrongly) that non-compliance is rife

-- when the damage potentially caused by non-compliance is high

-- when viable preventative mechanisms can be created

-- when some effective sanction can be applied post-hoc to those who do not comply

An optimistic enforcement does not involve preventative measures, but relies of detecting non-compliance and reporting/correcting them. This is widely used in real life. It tends to be used when:

-- when trust is high

-- when the potential damage due to non-compliance is low

-- when viable preventative mechanisms do not exist

The availability of viable preventative mechanisms has to be assessed against the objectives of the community. In real life, cheap convenient preventative mechanisms often exist but their use is prohibited by concerns about civil liberties. Or, to put it more simply, a community must weigh up the relative risk of non-compliance against the costs of enforcing compliance and the risks inherent in the compliance mechanism.

In establishing a strategy for enforcement it is important that the mechanisms do not attempt to enforce policy on a wider range of enterprise objects than the community has the authority to do.

7.5.6 Organisation of policy

TEMPORARY NOTE – The working group invites National Bodies to provide an explicit description of policy structure at an abstraction level possible and desirable for this purpose. Specifically, this description ought to state that a policy may apply to (collections of) objects, to (collections of) objects in roles, to collections of roles, or to collections of interactions, or to combinations of the above. It is also essential to distinguish between policy types, templates and instances in this context.

The Enterprise behaviour of a community is expressed in terms of the permitted pattern of interactions in which the objects fulfilling roles in the community can participate.

The specification of enterprise behaviour can be expressed by a series of sufficiently detailed predicates, each of which has to be satisfied before the corresponding interaction can take place. However, this approach leads to complex and ill-structured specifications. The specification task can be simplified if the specification is factored into a number of sub-models, each constraining a particular aspect of the behaviour.

An interaction is possible in the behaviour if it is allowed by each of the defined sub-models. Sub-models are identified in this Recommendation | International Standard to describe:

-- the behaviour required to satisfy the defining aspects of the community's purpose, in terms of required collections of actions, alternatives and allowable forms of concurrency.

-- limits on behaviour arising from the authorisation, or otherwise, of the enterprise objects to take part in interactions; this can be expressed by statements that the various objects have the capability to perform particular sets of actions.

-- a delegation sub-model expressing the degree to and circumstances in which one object can take over the role and responsibilities of another, acting as a substitute for it.

The second and third of these sub-models are discussed in greater detail below. Other sub-models may be constructed and used in enterprise specifications.

7.5.6.1 Permission rules

TEMPORARY NOTE – The working group notes that 'required permissions' is not defined. The working group invites National Bodies to provide a rewording of this subclause.

Each enterprise interaction involves a number of interacting objects each fulfilling particular roles involved in the interaction. Associated with each interaction is a set of required permissions, which must be present for the action to take place. Required permissions are either:

-- associated with a particular role in this interaction; or

-- associated with the interaction as a whole.

A required permission is associated with an object and possession of a required permission permits the object to satisfy part of the requirements for the interaction to take place. Thus an object is in possession of a required permission if it has been granted permission to perform the corresponding interaction or to fulfil a role involved in the interaction.

An interaction takes place when there is no other impediment if

-- for every required permission of the specified interaction that is associated with a role, the objects fulfilling that role have the required permission;

-- for every required general permission, an object in at least one of the roles involved in the interaction has the required permission.

Permission is granted by the passing of required permissions between objects. This passing of required permissions is itself an interaction and is, in general, subject to its own set of required permissions. Thus an object can pass a required permission on to other objects only if it holds a corresponding required permission for such required permission-passing.

A community specification must declare the minimum set of required permissions that must be associated with objects fulfilling its roles in order for its objectives to be achievable.

7.5.6.2 Delegation rules

TEMPORARY NOTE – The working group invites National Bodies to provide a rewrite of this subclause to provide an small set of abstract structuring rules.

A delegation graph defines whether one object can act on behalf of another with respect to the performance of a particular role or action_role. Each node in the graph represents an object and each edge in the graph is labelled with the operation subject to delegation and any specific constraints on the delegation.

Delegation graphs specify either relations between object types or relations between object instances.

An interaction can take place if each object fulfilling an action_role is either:

-- of the type specified in the basic behaviour; or

-- linked directly to an object capable of fulfilling the role by an arc in the delegation graph for which any constraints are satisfied.

7.6 [was 7.7] Enterprise object rules

An enterprise object is a model of an entity that is significant to the achievement of a community's objective. It may represent human beings, IT systems, or collections of these, or resources of various kinds. An enterprise object may be refined as a community at a greater level of detail. All enterprise objects in an enterprise specification fulfil at least one role in at least one community. In fulfilling their roles, enterprise objects perform, or are subject to, actions, some of which are interactions with other enterprise objects.

There is no hard and fast rule for determining which entities should be represented as enterprise objects in a specification. The choice rests with the specifier and it is made on the basis that an entity needs to be represented as having a separate identity in order to express some significant feature of the enterprise concerned. Entities that are not relevant to the problem in hand, or which can be abstracted as some quantitative measure are not represented as objects. Thus an object is needed where there are significant interactions or change of responsibility, but not if, for example, general stock level or capacity is relevant.

For example, in considering the trading of grain, batches of grain need not be objects when expressing management of the trading process, but become so when the test results from the individual batches become the subject of policy.

A special class of enterprise objects is that which is composed of objects known as Community Enterprise Objects (CEOs). This is an object that fulfils a role in some community, which, in that role, is further refined at a greater level of detail as some core community. The interactions between a CEO and its environment must map onto the interactions in the core community's specification between objects fulfilling environment roles and objects fulfilling core roles (which may be interface roles, see 7.5.2.3).

7.7 [was 7.8] Contracts

The concept of contract, as defined in ITU-T X.903 | ISO/IEC 10746-3, provides the means to specify actual communities. For a group of entities to be modelled as a community, there must be some implicit or explicit agreement about the group covering three things:

-- the objective for which the group exists;

-- the structure, policies and behaviour of its members;

-- the entities comprising its members.

Some aspects of this agreement (e.g. membership) may only apply to particular instances of a community type, while other aspects of the agreement may apply to all instances of a community type (and thus can be considered as part of the community type).

This agreement is modelled as the contract for the community. This contract specifies:

-- the objective of the community;

-- the community type which defines the behaviour and structure of the community in terms of:

    -- processes carried out by the community to fulfil the objective,

    -- roles, the relationship between the roles and the relationships of the roles to the processes,

-- policies that apply to the roles and actions of the roles;

-- the enterprise objects and/or types of enterprise objects comprising the community and the roles that they fulfil.

TEMPORARY NOTE - This subclause does not explicitly deal with types of contract and contract templates. Some subclause should. The working group invites National Bodies to propose text.

7.8 [new] Lifecycle of a community

TEMPORARY NOTE – The previous subclause 7.10, Evolution of a community, has been deleted, since no suitable text was offered. The working group notes that a community template does not change during the time it is used to instantiate communities.

Editor's note - Or: A change to a community template does not change communities previously instantiated from that template.

TEMPORARY NOTE – The following text was drafted during the Curitiba meeting as a proposal. The working group invites National Body comment.

The lifecycle of a community is closely linked with the lifecycle of objectives.

Communities, roles, and objects all can have objectives.

The starting point is the objectives of objects, when they have objectives. Where an object has an objective, the participation of a object in a community enables the object to meet its objective that would not be possible if the object acts individually. This impossibility could be due to physical or temporal constraints, limitations in the capability of the object, or lack of resources. Therefore, communities are established by objects (the parties) to achieve some shared sub-objective of those objects.

The contract for the establishment of the community will express how the community´s objective is to be achieved, by specifying the roles, processes, and policies of the community.

Objects filling roles must take on the objectives of those roles. Selecting an object to fill a role will typically include some assessment of the conflict between the objectives of the role and the objectives of the object (which includes the object's initial objectives and the objectives it has acquired by filling other roles).

NOTES:

1 – It is not generally possible to determine if objectives are in conflict. While it would be unlikely that an object fill a role which had an objective that was definitely in conflict with existing objectives, it is possible that an object will take on new objectives if the potential for conflict is assessed (at that time) as being low or if the object has a plan to manage conflicts in objectives when they occur (this may include the relinquishing of one or more roles and hence the objectives associated with that role).

2 – The distinction between an object that creates a community (one of its owners) and an object that merely fills a role in a community is of some importance. The object creating the community is committed to the objective of the community, whereas the object filling a role is committed to the objective of that role but not necessarily the wider objective of that community. Because the role's objectives may be a subset of the communities' objective, it is possible for an object to fill a role even though the objective of the community may be fundamentally in conflict with the object's own objectives. For example, an accountant in a company might be committed to the role's objective of an accurate set of company accounts but not to the company's objective of downsizing. This is the difference between "ends" and "means". Note that this is not the core/environment distinction, but a distinction between the parties to the contract and the fillers of core roles, best exemplified by the distinction between owners of a business (the parties to the contract), the employees of a business (core roles), and the customers of the business (environment roles).

7.8.1 Establishment of a community

A community is established by a contract, an agreement between parties (enterprise objects). The objective of the community will be a subset of the objectives of all of the parties of that community contract. The parties to the community contract are (collectively) the owner of the community upon instantiation.

The instantiation of a community can include the assignment of objects to roles within the community. The specification must state the minimum collection of roles required to instantiate the community, and whether the assignment of an object to a role is required to instantiate the role. For some roles, the lifetime of the role will begin with the assignment of an object and terminate when that object is de-assigned. Changing the assignment to another object, if permitted, will terminate the existing role and instantiate a new role. For other roles, the lifetime of the role will be independent of the assignment of objects to that role; the specification must state whether the role can exist when no object is assigned to it.

Editor's note - The term, 'role,' may be used in the previous paragraph sentence with a different meaning than that of [2-9.14]. The concepts, instantiation and termination, may not apply to roles that are an identifier for a behavior, or are used as a parameter in a template.

NOTE – There is no requirement that the parties fill roles within the community. Indeed, the lifetime of the community can exceed the lifetime of the parties to the community contract.

It is a modeller´s choice whether the instantion of the community will form part of the enterprise specification. That is, the establishing behaviour may be explicit in some specifications but not in others.

It is also the modeller´s choice whether the owner of a community is of relevance and, if so, whether the owner should be modelled as a community whose roles are filled by the parties to the community contract.

7.8.2 Changes in a community

A community can change in the following ways:

-- assigning and deassigning of objects to roles

-- creating and deleting roles (as instances)

TEMPORARY NOTE – What else might we want to consider changing in a community?

A specification must state the circumstances (if any) in which the above changes can occur during the lifetime of the community.

It is a modeller´s choice whether the changes to the community (for any of the above kinds of changes) will form part of the community specification. That is, community-changing behaviour may be explicit in some specifications but not in others.

7.8.3 Termination of a community

Communities can terminate in the following ways:

-- The community achieves its objective. This only applies when the objective was to reach a desired state (usually expressed as a predicate on the state) as opposed to objectives which seek to maintain or continually improve the future states (usually expressed in terms of weightings or thresholds calculated from the state).

-- The community fails because it cannot achieve its objective. This only applies when the objective includes a specification of the failure conditions.

-- By the decision of the owner of the community (which may or may not relate to the achievement or lack of achievement of the objective)

It is a modeller´s choice whether the termination of the community (for any of the above reasons) will form part of the community specification. That is, terminating behaviour may be explicit in some specifications but not in others.

One of the failure conditions for a community can include a minimum collection of roles that either must exist or must be filled by objects. Some communities will not be viable if certain roles do not exist or are unassigned (at any time or over some period of time).

7.9 [new] Force rules

TEMPORARY NOTE – The working group invites National Bodies to propose structuring rules for the force concepts of subclause 6.7.

7.10 [removed] Evolution of a community

TEMPORARY NOTE – The previous subclause 7.10, Evolution of a community, has been deleted, since no suitable text was offered. The working group notes that a community template does not change during the time it is used to instantiate communities.

Editor's note - Or: A change to a community template does not change communities previously instantiated from that template.

7.10 [new] Common community types

This subclause defines a number of common community types:

-- domain

-- federation

-- ownership community

7.10.1 Domain community type

An <X>-domain is a community type with one core role "<X>-controller" and one or more "<X>-controlled" roles, where the controller controls the controlled with regard to the <X>-aspect of their behaviour. Note. The core/environment nature of the controlled role is left for further refinement.

7.10.2 Federation community type

An <X>-federation is a community type with 2 or more core roles "<X>-federation member" which are filled by <X>-domains. The objective of an <X>-federation is to enable the control of the <X->controlled elements in the individual domains to be shared among the <X>-controllers of those domains. The specific manner in which the <X>-control is shared requires further refinement of the federation community type.

Note. At the level of abstraction at which federation is agreed, the federation members must be domains of the same type (X-controlling). However, each <X>-domain may actually be an instance of one or more refined domain types.

7.10.3 Ownership Community Type

Ownership is a subtype-by-specialisation of the domain community type. The object filling the controller role (also known as the owner role) must be the owner of the objects filling each of the controlled roles (also known as the owned roles). The controlled behaviour is all behaviour of the object filling the owned role apart from that which is controlled in other (non-subordinate) domains or which cannot be subject to control. Note. Since the owned roles may be filled by objects of many different types, the extent of control may vary among the owned objects of a common owner object."

7.11 [new] Scoping statement

Every enterprise specification shall have a scoping statement that defines the basic invariants that apply to that specification. A scoping statement says 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 so test conformance of observable properties to the specification.

Note: The provision of an accurate scoping statement is particularly important if reuse of the enterprise specification is expected. It allows the specifier who might incorporate the existing specification fragment to ask "is this specification for me?" before they begin to ask "what must my enterprise and its supporting systems do?".

8 Framework for the specification of policy

TEMPORARY NOTE - This clause covers material similar to that in clause 7. The working group invites National Bodies to comment.

Editor's note - This clause uses a term, principal, for a concept which is different from the principal of subclause 6.6.3.

Within a community constraints are placed on the behaviour of objects fulfilling principal roles in that community in the form of policies. It is here that the environment in which the community is specified becomes particularly important. No community can operate in isolation from its context; it cannot arbitrarily place or relax policies, but is itself constrained by the situation which it inherits from the environment in which it is created and the obligations taken or responsibilities devolved by its principals.

In principle, there is an unconstrained outer level for specification in which there are no specific policies, and any behavioural interaction definable in terms of the underlying object model may be attempted, but may be arbitrarily rejected by other participants in the interaction.

In practice, this ideal leaves too large a specification task to be completed, and realistic specifications will start by declaring an assumed environment in terms of outer communities, or by enumeration of fixed policies.

Some of the ways in which creation of a community changes the policies that apply to its members are:

a) the principals agree, by participation in the behaviour which creates the community or introduces them to it, to be subject to policies; their behaviour is then more restricted than it was before;

b) one of the principals may, by virtue of its role in some outer community, invest the community with responsibility to impose policies on arbitrary members which are not principals; this implies that the acceptance of such policies from proper authority was already prescribed in the outer community.

c) the principals may undertake obligations not to do things unless specifically allowed by the rules of the community; this gives rise to a structure of permissions, administrated by some role in the community;

d) a structure for the delegation of permissions may be constructed in the same way as given in (b) for obligations.

Note – Permission is used here in the sense of "have permission". The binary relation "give permission" also implies an obligation on the giver to enable, and not to obstruct, the permitted action, but the details of this require further study.

The above implies that every principal role identifies behaviour that is concerned with the fulfilment of some policy that is directly related to the achievement of the objective of the community.

9 Conformance and reference points

An enterprise specification conformant with this Recommendation | International Standard shall use the concepts defined in clause 6 and those in ITU-T Rec. X.902 | ISO/IEC 10746-2 clause 5.1, structured as specified in clause 7.

Concepts from ITU-T Rec. X.902 | ISO/IEC 10746-2 not refined in this Recommendation | International Standard may also be employed. Where such concepts are employed, the specification concerned shall include unambiguous explanations of the relationships between the concepts concerned and those defined in Clause 6.

Editor's Note - National Bodies are invited to propose other wording for the immediately previous sentence.

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, ITU-T Rec. X.902 | ISO/IEC 10746-2, or ITU-T Rec. X.902 | ISO/IEC 10746-2 clause 5.1, and unambiguous explanations of the relationships between such concepts and those defined in clause 6.

Editor's Note - The editor was instructed, but has failed to rewrite the immediately previous sentence. National Bodies are invited to propose other wording.

[See also 3-5.3]

10 Consistency rules

TEMPORARY NOTE – The working group invites National Body comment and contributions on consistency rules. The working group intends:

-- If consensus can be reached that a statement in clause 10 is correct and belongs as a normative statement, it will be restated in the style of Part 3.

-- If consensus cannot be reached that a statement in clause 10 is correct and belongs as a normative statement, it will be removed.

-- The non-normative, explanatory and illustrative material will be removed.

TEMPORARY NOTE – The working group considers that, due to changes in terms and definitions, some of these correspondences are incorrect.

TEMPORARY NOTE – The working group are not agreed that the diagrams in this clause 10 are correct.

A set of specifications of an ODP system written in different viewpoint languages should not make mutually contradictory statements (see 3-4.2.2), i.e., they should be mutually consistent. Thus, a complete specification of a system includes statements of correspondences between terms and language constructs relating one viewpoint specification to another viewpoint specification, showing that the consistency requirement is met. The minimum requirements for consistency in a set of specifications for an ODP system is that they should exhibit the correspondences defined in ITU-T Recommendation X.903 | ISO/IEC 10746-3 and those defined within the set of specifications itself. ITU-T Recommendation X.903 | ISO/IEC 10746-3 does not declare generic correspondences between every pair of viewpoint languages. This clause is restricted to the specification of correspondences between an enterprise specification and other viewpoint specifications. In each case, the correspondences are expressed as interpretation relationships linking terms in one viewpoint language to terms in the other viewpoint language. A set of specifications based on this Reference Model will, in general, need to relate all the 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 specifications in a second specification. Correspondences can be established between two different specifications in a single language or in two different languages. Statements of correspondences between two languages imply equivalent correspondences between any pair of specifications expressed in those languages.

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 a entity needs to be consistent with what is said about the same entity in any other viewpoint specification in its representation about that entity's properties, structure and behaviour.

A specification of an ODP system written in the enterprise viewpoint language should not make statements which are contradictory (see ITU-T Recommendation 903 | ISO/IEC 10746-3 subclause 4.2.2) with statements in specifications written in other viewpoint languages, i.e., the specifications should be mutually consistent. A complete specification of a system includes statements of correspondences between terms and language constructs relating one viewpoint specification to another viewpoint specification, showing that the consistency requirement is met. 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, those defined in this standard, and those defined within the set of specifications itself.

The correspondences between instances of concepts in an enterprise viewpoint specification of a system and instances of concepts in the information, computational and engineering languages are summarised in Figures 10-1, 10-2 and 10-3, using a UML notation. Details of each type of correspondence are given in Subclauses 10.1, 10.2 and 10.3.

NOTE – 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.

10.1 Enterprise and information specification correspondences

It is your project editor's opinion that this is related to the concept of reference points, and he will produce text expanding on this notion in advance of the next meeting.

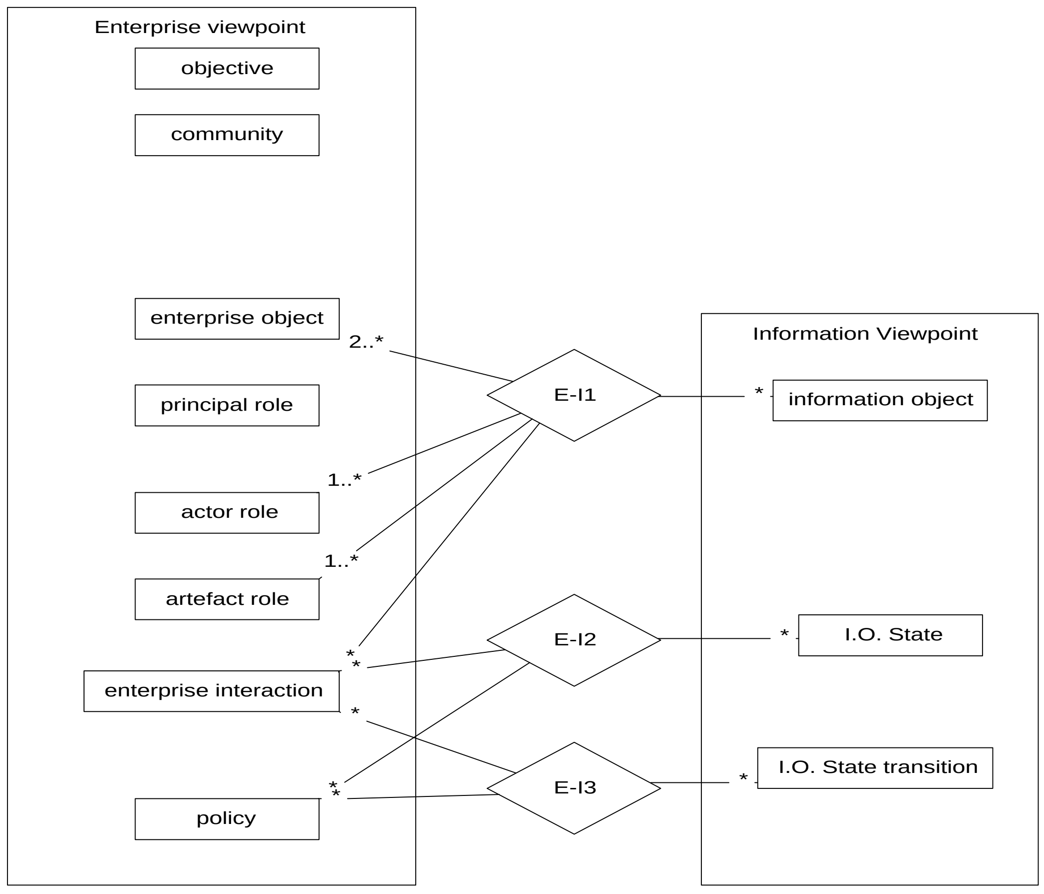

Figure 10- 1 Correspondences between the enterprise viewpoint and the information viewpoint

For the purpose of viewpoint consistency, the key elements in the information viewpoint are information objects, and relationships (i.e. relationships between information objects), which are modelled as invariant schemata; static schemata; and dynamic schemata.

Note: In some notations and at some level of abstraction, information objects comprising a composite information object may be represented as attributes.

The following generic correspondences between instances of concepts in the enterprise language and in the information language will exist in any consistent set of viewpoint models of a system:

E-I1 An Information Object may represent the information content about an enterprise object fulfilling any kind of role where that enterprise object is the subject of an enterprise interaction that is part of the behaviour of the system in one of its actor roles.

E-I2 Allowable states of information objects representing enterprise objects may be governed by policies in the enterprise description

E-I3 Allowable state transitions of information objects representing enterprise objects may be governed by policies in the enterprise description

10.2 Enterprise and computational specification correspondences

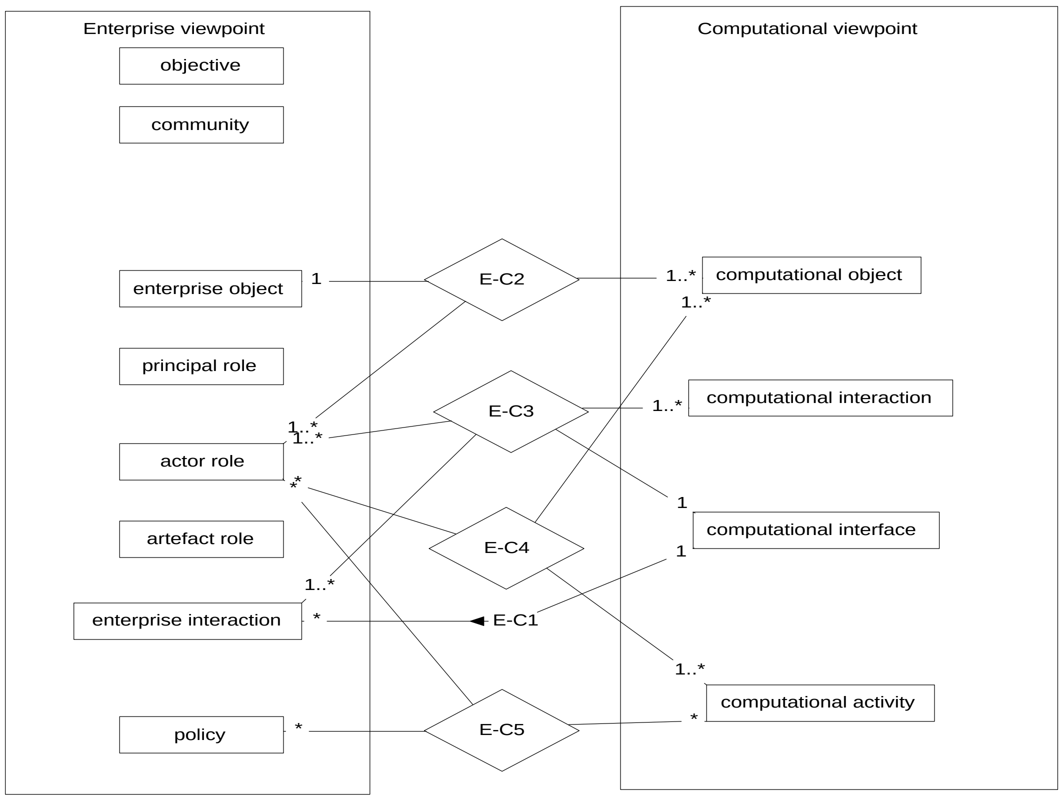

Figure 10- 2 Correspondences between the enterprise viewpoint and the computational viewpoint

For the purpose of viewpoint consistency, the key elements in the computational viewpoint are computational objects, computational interfaces, interactions between computational objects at their computational interfaces, and activities. A computational interface may support a number of types of enterprise interactions. A computational object may have a number of computational interfaces with definitions of constraints on the behaviour at those interfaces.

The following generic correspondences between instances of concepts in the enterprise language and in the computational language will exist in any consistent set of viewpoint models of a system:

E-C1 A system (enterprise object) in one of its roles in the enterprise viewpoint specification for the system corresponds to a computational object or a configuration of computational objects in the computational specification.

E-C2 An interaction of the system in one of its roles corresponds to one or more interactions at one or more interfaces of the corresponding computational object or configuration of computational objects.

E-C3 The behaviour of a system in one of its roles corresponds to one or more activities of the corresponding computational object or configuration of computational objects.

E-C4 A computational activity that is part of the behaviour of an actor role must be conformant with any policies that govern that role.

E-C5 A computational activity that is part of the behaviour of an actor role must be conformant with any policies that govern that role.

10.3 Enterprise and engineering specification correspondences

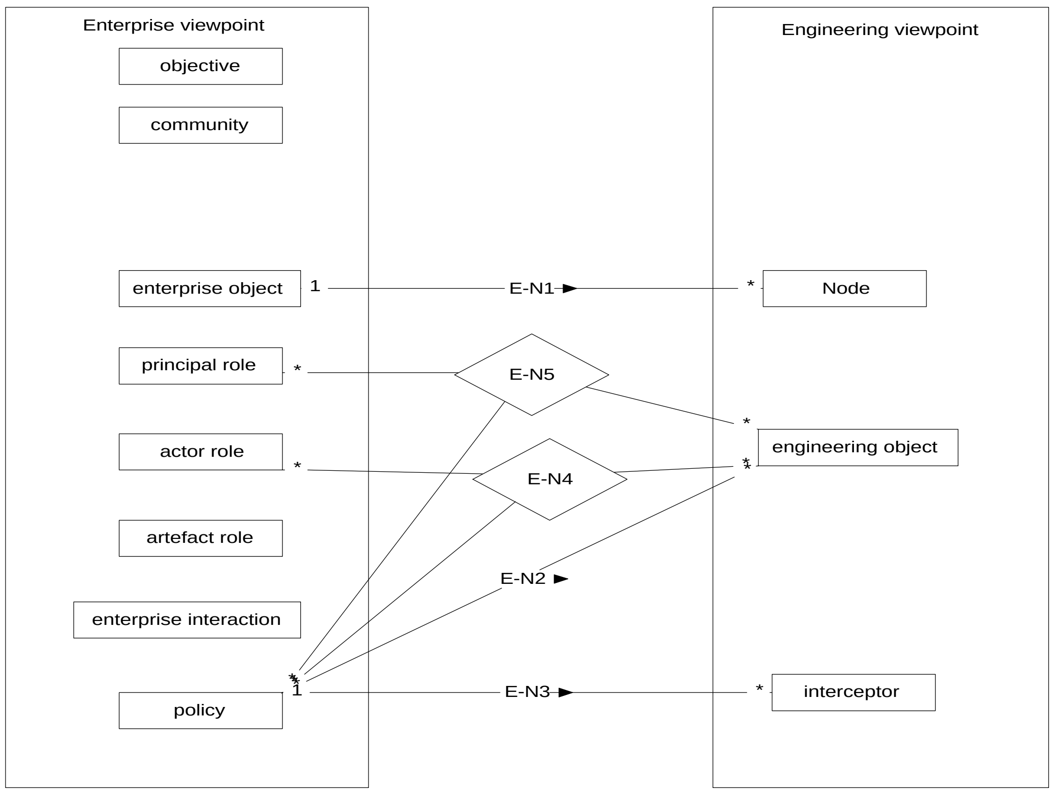

Figure 10- 3 Correspondences between the enterprise viewpoint and the engineering viewpoint

The following generic correspondences between instances of concepts in the enterprise language and in the engineering language will exist in any consistent set of viewpoint models of a system:

E-N1 The location of an engineering node is determined by location of an enterprise object with which it corresponds.

E-N2 Enterprise policy may determine choice of transparency mechanisms and supporting engineering objects.

E-N3 An enterprise policy may correspond to an engineering interceptor which implements it, at an administrative domain boundary, or otherwise.

E-N4 An actor role may be fulfilled by an engineering supporting object where the role concerned is essentially a policy enforcement role, and the enforcement process is automated within the system.

E-N5 A principal role may be fulfilled by an engineering supporting object where the role concerned is essentially a policy enforcement role, and the enforcement process is automated within the system.

Drafter’s Note: The drafter is not aware of any circumstances where a principal role that is to do with policy enforcement could also be an artefact role, and he believes that E-N5 is therefore redundant. This view is not shared by all UK experts, and it is felt that this should be discussed at the Brisbane meeting.

Annex A Overall structure of an enterprise specification

The relationships between the concepts used in an enterprise specification are illustrated in Figure A-1.

> Note: The diagram is not normative and nor is it a stand-alone representation. It shows the more significant relationships and should be read in conjunction with the normative text in this Recommendation | International Standard.

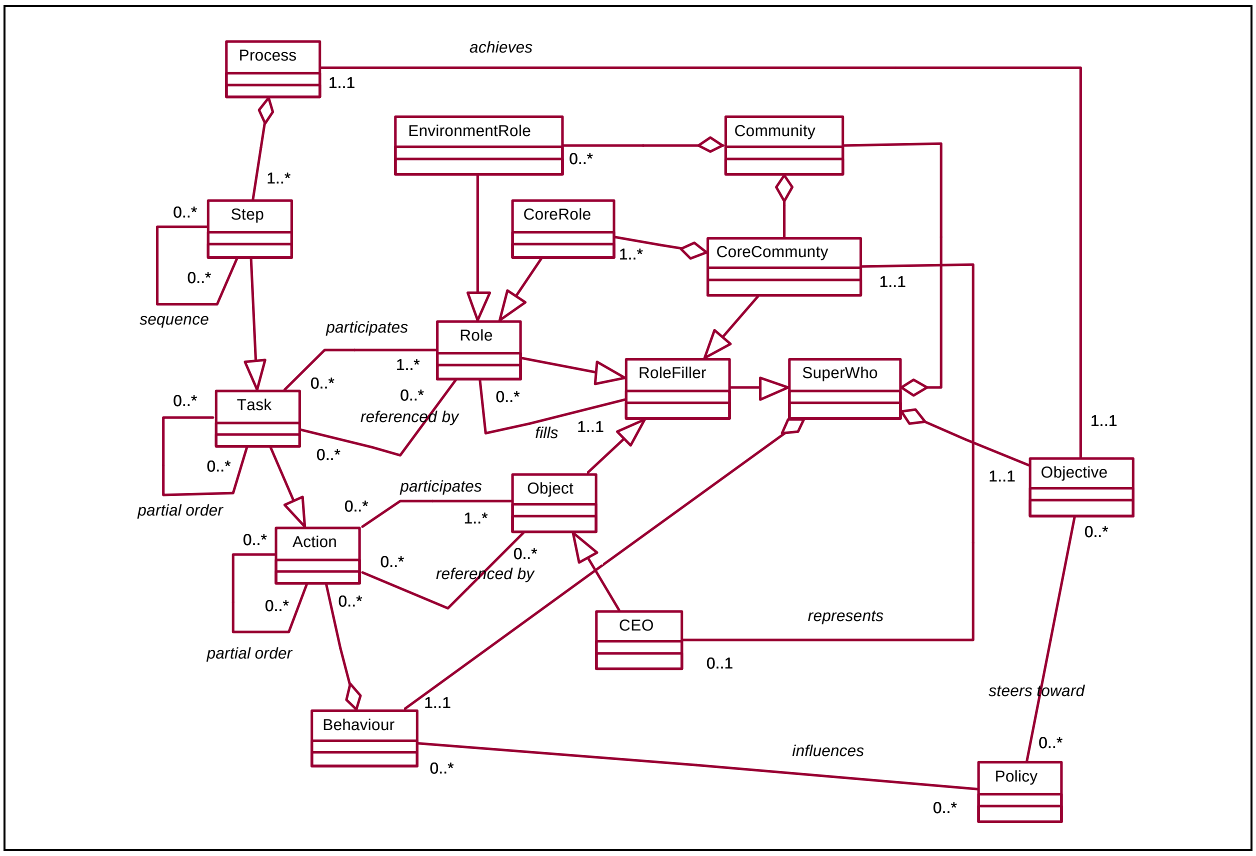

Figure A-1 Working language model

An enterprise specification will include of a Scoping Statement, one or more specifications of community types and zero or more community population statements . The Scoping Statement says 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 so test conformance of observable properties to the specification.

Note: The provision of an accurate scoping statement is particularly important if reuse of the enterprise specification is expected. It allows the specifier who might incorporate the existing specification fragment to ask "is this specification for me?" before asking "what must my enterprise and its supporting systems do?".

Each community type definition will contain information about:

-- optionally, the behaviour (processes) of the business or social organisation for which the community is formed

-- the roles, constraints on roles and relationships between roles that, collectively, identify the behaviour of the community

-- the policies that govern behaviour associated with roles and assignment of enterprise objects to roles

-- relationships between instances of all the above concepts.

In addition, depending on how concrete the community being described is, a community type specification can describe or identify:

-- the objective for which the community is formed

-- the population of enterprise objects or types of enterprise object that comprise the community and the assignment of objects to roles.

In general a community type specification will form part of a set of community type specifications which may be related to one another in a variety of ways, in each case with corresponding cross references between the specifications concerned:

-- one community type may be a refinement of another – this is a case of reuse of community type specifications;

-- a community type may be the specification for a community that is a refinement of a single enterprise object, referred to in Figure A-1 as the CEO or Community Equivalent Object.

The basic structure is illustrated using an example textual notation in Figure A-2

An example of the basic structure of an enterprise specification is illustrated in the Figure A-2 below using extended BNF notation. The syntax of the notation used is explained in Figure A-3.

TEMPORARY NOTE - The working group is aware that this illustration is not fully correct, but is included to illustrate the point that the structure of an enterprise specification can be framed in such terms. The working group invites National Body comments and suggestions.

1. <enterprise specification> ::= <community>+
2. <community> ::= <community tag> [<community name>] [<refines statement>] <behaviour> <objective> < population assignment>*
3. <refines statement> ::= <refines tag> <community name> <refines specification>
4. <behaviour> ::= <process>* < role>+ <policy>*
5. <process> ::= <process tag> <process name> <process specification>
6. <process specification> ::= <step>+
7. <step> ::= <step name> <task> <sequence statement>
8. <sequence statement> ::= <sequence tag> <step name>
9. < role> ::= <environment role tag> <role name> <task>+ | <core role tag> <role name> <task>+
10. <task> ::= <action>
11. <action> ::= <action name> <action specification> <constraint>* <involves statement>* <references statement>*
12. <constraint> ::= <constraint tag> <constraint specification>
13. <involves statement> ::= <involves tag> <actor name>+
14. <references statement> ::= <references tag> <artefact name>+
15. <policy> ::= <policy tag> <policy statement>
16. <policy statement> ::= <policy specification> <applies tag> <process name> | <policy specification> <applies tag> <role name>
17. <objective> ::= <objective tag> <objective statement>
18. <population assignment> ::= <assignment tag> <role-object assignment>
19. <role-object assignment> ::= <role name> <assigned tag> <object>
20. <object> ::= <object tag> <object name> <object description>

Figure A-2 BNF Representation of the structure of an enterprise specification

Notes on Figure A-2

The unspecified non-terminals other than tags and names (which are all syntactically "identifier") are:

<refines specification>
<action specification>
<constraint specification>
<policy specification>
<objective statement>
<object description>

Symbol	Meaning
::=	Is defined to be
|	Alternatively

<text>	Non-terminal
"text"	Literal
*	The preceding syntactic unit can be repeated zero or more times
+	The preceding syntactic unit can be repeated one or more times
{}	The enclosed syntactic units are grouped as a single syntactic unit
[]	The enclosed syntactic unit is optional (may occur zero or one time)

1 Figure A-3 Extended BNF Notation

Annex B ODP system rules

In an enterprise specification, an ODP system and the environment in which it operates are represented as a community. At some level of description the ODP system is represented as an enterprise object in the community. These two crucial sentences explain the purpose and compass of an enterprise specification.

The focus of an enterprise specification conformant with this Recommendation | International Standard shall be the purpose, scope and policies of the ODP System (as defined in ITU-T X.902 | ISO/IEC 10746-2) being specified.

The purpose of the system and the objective(s) of the community or communities of which it forms a part must be consistent.

The scope of the system, where the term scope has the meaning defined in clause 6 above, will be the necessary and sufficient set of statements about its behaviour, such that information, computational, engineering and technology specifications can be developed.

NOTE – In order to understand that behaviour, it may be necessary to model at both more abstract and more detailed levels of description than that at which "the ODP system is represented as an enterprise object in the community." Thus this Recommendation | International Standard makes no prescriptions about either the most detailed or the most abstract levels of any enterprise specification, nor does it make any recommendations about the relative merits of modelling from 'top-down' or 'bottom-up'. The approach taken will be a modelling choice based on the system being specified and the purpose of the modelling.

The policies of the system are the set of policies of the organisation in which the system operates that apply to the system's behaviour.

Drafter's Note: The above statement limits policies to behaviour. This is in order to exclude from the enterprise viewpoint statements of policy such as "We always buy from Vendor <x>" Are there any non-behavioural policies that SHOULD be included in the enterprise viewpoint?

Text parking lot

Text here is not part of the committee draft. It is included for the convenience of the working group.

Text formerly at:

7.6.3 Nesting of policy frameworks

In the absence of a policy framework of an outer community, the existence of a policy framework for some behaviour in a core community implies that that behaviour is totally constrained by the core community's policies. But this core community policy framework does not bind non-members of the community, and other mechanisms will be needed to impose the compliance of non-members (if so desired). For example, an environmentally conscious group might decide in which circumstances members are permitted to drive their cars. While group members may be bound by these policies as a condition of membership of the group, this community is unable to constrain the (ab)use of cars by non-members through policy alone.

In the presence of a policy framework of an outer community, the policy framework of the core community is bounded by the framework of the outer community. The core community cannot permit what is prohibited by the outer community, unless it has a delegated authority to do so. The core community may be able to prohibit or obligate its members, provided the outer community does not prohibit the core community from imposing such policies. For example, a national driving licence authority might be permitted to decide the rules for the issuing of drivers licences (e.g. requiring some test of driving competency), but the government might obligate it not to issue drivers licences to applicants under 18 years old. That is, the authority cannot give a driver's licence to a child, even if the child can pass the competency test.

The way that non-members of the community are constrained may only be derived from some outer level community which does involve all the parties concerned. Therefore, policies made by the core community are respected in the outer community, only when the authority of the core community is obtained by delegation from the outer community.

An enterprise object (especially one filling roles in multiple communities) can acquire a collection of permissions and prohibitions with overlapping scope (e.g. pertaining to the same participant-role). A superior authority must provide a resolution mechanism to determine whether a given collection of permissions and prohibitions does or doesn't make the enterprise object willing to perform the participant-role. A resolution mechanism of a delegated authority may be constrained by the nature of its delegation from the superior authority.

Index