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

ISO/IEC JTC1/SC7 N2252

2000/01/28

Document Type	CD Ballot
Title	CD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint.
Source	JTC1/SC7 Secretariat
Project	07.77
Status	2nd CD Ballot
References	N2250
Action ID	FYI or ACT
Due Date	2000/04/28
Mailing Date	2000/01/28
Distribution	SC7_AG
Medium	Encoded Acrobat
No. of Pages	46
Note	Joint project with ITU-T (ITU-T X911).

ISO/IEC JTC1/SC7 CD 15414	
Date 2000/01/28	Reference number ISO/JTC 1/SC 7 N2252
Supersedes document **N2187**	

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) **2000/04/28** 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: 46

Vote on CD 15414	
Date of circulation **2000/01/28** Closing date 2000/04/28	Reference number ISO/JTC 1/SC 7 N2252

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

ISO/IEC JTC1/SC7/WG17 W17N0106

ITU-T X.911 ISO/IEC 15414

Date: 00-01-14

Committee Draft
Paris 1999 Output

14 January 2000

ISO/IEC JTC 1/SC 7

Software Engineering

Secretariat: Canada (SCC)

Doc Type:	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 committee draft; output from November 1999 Paris meeting
Action:	For CD ballot
Distribution:	SC 7
Medium:	E
Number of Pages:	45
Version:	202

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 **X.911**

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(Committee Draft 1/00)

COMMON TEXT DRAFT

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

ITU-T Recommendation X.911

Version 202

CONTENTS

Page

Foreword

This is the second 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 November 1999 Paris 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 may change depending on the software or 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

- Font

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; 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

This annex is 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]

TEMPORARY NOTE – Canada has made this comment: "The target audience of the document should be clearly stated in clause 1." See the temporary note at the beginning of clause 5.

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.

Identical ITU-T Recommendations | International Standards

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

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

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

3 Definitions

3.1 Definitions from ODP standards

3.1.1 Modelling concept definitions

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

-- community;

-- computational interface;

-- dynamic schema;

-- <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 – Canada has made this comment: "It is not clear if the target audience of this document is. This can affect the style of the document.

"If the target audience is those people who design languages for such type of specification, then the style has to be quite formal, and the normative part of the standard rigourous enough to enable conformance verification.

"If the target audience is those who prepare or use such specification, then the nature of the document is more like a guideline, and the style is less formal.

"The target audience of the document should be clearly stated in clause 1, and the requirements from this target audience stated in clause 5."

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.

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.

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

Editor's Note – Do we mean to suggest that the enterprise language is intended or best suited for formal specification?

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 environment with which it interacts. Such an environment can be a technical environment (e.g., the software and hardware environment of a service component) or a social or business organisation (e.g., a group of co-operating companies, a particular service inside a company).

The enterprise language defines the concepts necessary to represent the behaviour expected of an ODP system. It defines structuring rules for using those concepts to produce an enterprise specification.

An enterprise specification of an ODP system is an abstraction of the system and a larger environment of which the ODP system forms a part, describing those aspects that are relevant to specifying what the system is expected to do in the context of purpose, scope and policies of that environment (technical, organisational). It describes the behaviour assumed by those who interact with the ODP system. It explicitly includes those aspects of that environment 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.

The motivation for a standard enterprise language is to support standardized techniques for specification. This improves communication and consequently helps understanding.

TEMPORARY NOTE – The working group invites National Body contributions expanding on the benefits mentioned in the previous paragraph.

6 Concepts

The concepts of the 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.

6.1 General concepts

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

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

NOTE – In actual specifications, it is important to clarify a distinction between prescriptive postconditions for actions and those postconditions which include preferences about particular outcome of an action. The former are not objectives and the latter are. The details of the clarification may depend on actual specifications.

6.1.3 Scope (of a system): The behaviour identified 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.

6.1.5 <S>-community: A community in which an ODP system is represented as a single enterprise object interacting with its environment

Editor's Note - The preceding text is the same as the relevant text from clause 7.1. It is used in this committee draft to maintain consistency with clause 7.

The following text, proposed by the editor, follows Part 3. It allows for different models of the same <s>-community, with the system appearing as a single composite object in some models and as its component objects in others, as provided by Part 3.

6.1.5 <S>-community: A community consisting of an ODP system and the environment in which in which it operates.

NOTE - At some level of description the ODP system is represented as a single enterprise object in the <s>-community.

TEMPORARY NOTE – It is not the intent of the working group that the term '<s>-community' should appear in this Recommendation | International Standard. This term is a placeholder for an acceptable term. The working group invites National Body contributions.

6.1.6 <C>-object: A composite enterprise object used to represent a community when specifying the participation of that community in another community. The decomposition of a <c>**-**object is a configuration of some of the objects in the community represented.

TEMPORARY NOTE – It is not the intent of the working group that the term '<c>-object' should appear in this Recommendation | International Standard. This term is a placeholder for an acceptable term. The working group invites National Body contributions.

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

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

6.1.8 Owner (of an <X>): The party or one of several parties having the right to control the use and disposal of the <x>.

NOTES

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

2 – An owner is thus an enterprise object modelling an entity that is an actual owner of the entity modelled by the <x>.

Editor's Note - Part 2 uses '<X>' in both subclause headings and in body text. Part 3 (mostly) uses '<X>' in subclause headings and <x> in body text. This draft uses <x>, <s> and <c> in body text. The editor invites National Body comment on this question of style.

6.2 Behaviour concepts

6.2.1 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 starting points.

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

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

6.2.2 Task: An action in the behaviour of a community. Tasks are associated with roles.

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

Editor's Note **–** To accommodate a comment from one of the working group, the editor has removed the quotations marks which appeared around the term, task, in the initial committee draft.

6.2.3 Step: A task in a process.

TEMPORARY NOTE – This draft may use more concepts than needed. The working group considers the choice of terms here a major issue. The working group invites National

Body contributions proposing a minimal set of concepts (from action, activity, task, step, or others).

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

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

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

6.2.6 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.

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

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

6.2.9 Resource role (with respect to a community): A role in which the enterprise object filling the role is involved in any actions of the role only as a resource.

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

6.2.10 Interface role (with respect to a community):

TEMPORARY NOTE – The working group invites National Body suggestions for a definition of this role concept in accordance with the text of 7.7.2.1

6.2.11 Contracting party (with respect to a contract): A party modelling an entity that agrees to that contract.

6.3 Policy concepts

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

NOTES:

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

2 – This definition refines 2-11.2.7.

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

NOTE – Unlike a permission, an authorisation is an empowerment

6.3.3 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.

6.4 Delegation concepts

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

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

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

NOTES –

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

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

TEMPORARY NOTES –

1) This term is intended to have a meaning that follows one standard meaning of ‘agent,’ that in the pair agent/principal. This meaning may be different from a currently popular use of ‘agent,’ which may be closer to that in the pair agent/patient. The working group invites National Body comment on this question and, if considered necessary or appropriate, suggestions for a different term.

2) The working group invites National Body comment on structuring rules using this concept.

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

6.5 Force concepts

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

These concepts may be used to model changes in that part of the universe of discourse modeled by the environment of the ODP system, including such changes when caused by changes in the ODP system itself.

6.5.1 Act: An action in which one or more objects is a party or agent and which action is a model of something done by an entity modelled by a party or its agent.

6.5.2 Commitment: An act by which the entity represented by a party is bound by a rule or by a contract. A commitment creates an obligation to comply with the rule or perform the contract.

NOTE – The enterprise object(s) participating in an act of commitment may be parties or agents acting on behalf of a party. In the case of acts of an agent, it is the entity modelled by the principal which in bound.

6.5.3 Declaration: An act that creates a state in the universe of discourse modelled by the environment of the ODP 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.

6.5.4 Delegation: An act that delegates (authority, a function, etc.)

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

6.5.6 Evaluation: An act that assigns a value to something.

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

6.5.7 Instruction: An act that is intended to cause a person or machine to do something.

6.5.8 Prescription: An act that establishes a rule.

7 Structuring Rules

7.1 Overall structure of an enterprise specification

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

A fundamental structuring concept for enterprise specifications is that of community. A community is a configuration of enterprise objects modelling a collection of entities (e.g. human beings, information processing systems, resources of various kinds and collections of these) that are subject to some implicit or explicit contract governing their collective behaviour.

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

- roles played by the ODP system;
- activities undertaken by the ODP system within processes in which it participates;
- policy statements about the system, including those relating to environment contracts.

An enterprise specification of an ODP system includes at least a description of the community in which that system is represented as a single enterprise object interacting with its environment. This is referred to as the <s>-community.

NOTE – This minimal enterprise specification details the objective and scope of the ODP system and is necessary for completeness of the enterprise specification.

TEMPORARY NOTE – It is not the intent of the working group that the term '<s>-community' should appear in this Recommendation | International Standard. This term is a placeholder for an acceptable term. The working group invites National Body contributions.

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

NOTE – In order to understand the ODP system behaviour, it may be necessary to describe communities at both more abstract and more detailed levels than the minimal enterprise specification. These communities may have objectives that diverge from, or contradict, the purpose of the ODP system itself.

The enterprise specification can also be structured in terms of a number of communities interacting with each other. In such a case the communities concerned, viewed as composite objects (<c>-objects), themselves form a community that may be described explicitly or may be left implicit and the ODP system must be a member of at least one of those communities

NOTE – Such structuring may represent, for example, co-operation of domains in a federation.

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 the behaviour of the ODP system such that information, computational, engineering and technology specifications can be developed.

NOTE – 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 is a modelling choice based on the ODP system being specified and the purpose of the modelling.

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.

7.2 Scoping statement

Every enterprise specification has 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?”.

7.3 Community rules

7.3.1 Specification of a community

For a collection of entities to be modelled as a community there must be some implicit or explicit agreement about the collection covering the reason for its existence, how it is organised and what it does, and what entities comprise its membership. This agreement is expressed as the contract for the community that

- covers the objective for which the community exists,
- governs the structure, the behaviour and the policies of the community,
- constrains the behaviour of the members of the community,
- associates enterprise objects with roles as members of the community.

The contract can be formed either by a defined process carried out by some or all of the enterprise objects at the time of community establishment, or in an earlier epoch, for example, as a design decision.

NOTES

1 – The concept, contract, is defined in ITU-T X.902 | ISO/IEC 10746-2.

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

The behaviour of the community realises its objective. In the context of the community, the objectives of members of the community are constrained to conform to its objective.

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

- the processes that take place in the community;
- any interactions of the community and its environment;
- the relationships between the processes and any interactions that exist;
- policies that apply to the processes and to any interactions that exist.

A behaviour of a community is composed of actions that are identified by the roles of the community. Constraints on these actions must be consistent with the constraints identified by the roles of the community.

The behaviours of objects in a community are subject to the contract of that community.

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

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

NOTES

1 – Types of communities or community templates may be used in the description of these communities.

2 – Types of communities may be related by refinement.

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

7.3.2 Relationships between communities

An enterprise specification can describe several communities. A community can 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 enterprise specification prescribes that the enterprise object fulfilling a role in one community be part of another community, perhaps fulfilling a certain role in that community.

-- A composite enterprise object is part of a community and it or one of its component objects is part of another community.

NOTES:

1 – The first case occurs where 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 occurs where an outer community 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.

2 – The second case is where it is necessary to consider the community of interest (perhaps one in which the ODP system appears as a single enterprise 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.

Alternatively, 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.

3 – A specification may prescribe, as a part of the specification of a community, that a certain enterprise object in the community is a composite object represented, at a different level of abstraction, as another community. Or it may happen that an enterprise 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 enterprise objects in some other, otherwise unrelated community.

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

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 the objects of this outer, English Legal community. By being defined within this context, the community representing the company, by virtue of fulfilling a specific role in the English Legal community, acquires a new type, i.e., a corresponding set of obligations on its members, representing the requirement that they operate in accordance with the law. This may call for conflict determination and resolution mechanisms to be used for acquiring such new types.

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.

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

7.4 Common community types

Three community types are:

-- domain

-- federation

-- ownership community

7.4.1 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.

7.4.2 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.

7.4.3 Ownership Community Type

The enterprise object filling the controller role (also known as the owner role) must be the owner of the enterprise 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 domains or which cannot be subject to control.

Note - Since the owned roles may be filled by enterprise objects of many different types, the extent of control may vary among the owned objects of a common owner object.”

7.5 Lifecycle of a community

7.5.1 Establishment of a community

A community is populated by assigning an enterprise objects to roles of the community. The behaviour of the enterprise object assigned to a role must be consistent with the behaviour identified by the role. Furthermore, the enterprise object selected to play a certain role must fulfil the requirements of the population policy.

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

The enterprise objects assigned to roles in the community can be dynamically changed during the lifetime of the community. As a consequence, a role can temporarily be empty. Still, the community is continuously responsible for the obligations placed on that role.

Editor's Note - See last editor's note in 7.9, Enterprise object rules, regarding text that was moved from this subclause to 7.9.

7.5.2 Changes in a community

A community can change in the following ways:

-- assigning and deassigning of objects to roles

-- creating and deleting roles

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

Editor's Note – The form 'deassign' certainly is authorized by OED: "de-, prefix II. As a living prefix, with privative force. 1. Forming compound verbs … having the sense of undoing the action of the simple verb…" Yet it seems startling and awkward to this reader. The OED does suggest: "(The hyphen is conveniently used … to emphasize the occasional nature of the combination…" That gives: assigning and de-assigning of objects to roles.

A specification must state the circumstances (if any) in which such changes can occur during the lifetime of the community. If a role is to be created, the specification of the community must include a specification of the type of the role.

Editor's Note – The instructions from the Paris meeting call for the use of the concept, type of role, here. Note that Part 2 might be read as teaching us instead to use role template here.

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.

7.5.3 Termination of a community

Editor's Note – The term 'termination' is not a foundation concept from Part 2 and has a technical meaning in Part 3. If this concept is to remain here, a different term might be chosen.

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 enterprise 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).

7.6 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 a composition of sub-objectives.

NOTE – The objective of the community is not necessarily a reachable goal but may be an aspiration towards some less specific aim.

Objectives of a community can be expressed by a collection of interrelated roles, processes, and policies. Objectives expressed in a role are realised by objects fulfilling the role. Objectives expressed in a process are realised by objects performing the actions of the process. 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 each different sub-objective. A policy may influence the occurrence of a specific action in a process, which may in turn result in occurrences of other actions as well as initiation or termination of processes.

An enterprise object can have objectives where the entity being modelled has objectives.

An enterprise object fulfilling a role has the objective of the role. Hence, where a community, viewed as a composite object, fulfils a role in a community of wider scope, the objective of the first community is consistent with the objective of the role.

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

7.7 Behaviour rules

Editor's Note – To accommodate a comment from one of the working group, the editor has, throughout this subclause, removed the quotations marks which appeared around the term, task, in the initial committee draft.

7.7.1 Roles and processes

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”.

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

NOTES:

1 – The behaviour of a community includes interactions of the community, viewed as a composite object, with its environment.

2 – There are many specification styles for expressing the ordering of “tasks.” The modelling language chosen for expressing an enterprise specification may impose certain styles.

The assignment of tasks to the enterprise objects that comprise a community is defined in terms of roles. A role identifies an abstraction of the community behaviour that comprises a set of tasks that is assigned to a single enterprise object within the community. Each abstraction is labelled as a role. The emphasis is on which enterprise objects participate in the particular behaviour.

The tasks and their ordering are defined in terms of processes. A process identifies an abstraction of the community behaviour that includes only those tasks that 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 enterprise object in the community. The enterprise 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.

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 enterprise 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.

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. As a minimum the roles of the enterprise 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 7.7.4.

7.7.2 Role rules

In the specification of a community, each role stands as a placeholder for some enterprise object that exhibits the behaviour identified by the role.

In an enterprise specification the term ‘<x> object’, where <x> is a role, is interpreted as meaning an enterprise object fulfilling an <x> role. Where an enterprise object fulfils multiple roles, the names can be concatenated

An enterprise object may fulfil several roles in one community, and may fulfil roles in several communities. Any role in a community may be filled by different objects at different times, but only by one enterprise object at any one time. At any location in time a role in a community may be unfilled.

When a community template is instantiated, at most one enterprise object is associated with each role. The constraints of the behaviour named by the role become constraints on the object fulfilling the role.

Editor's Note - These constraints also become constraints on other objects later fulfilling that role. If this is implied by other role rules, then the previous sentence may not be necessary.

When an enterprise object is assigned to a role the types of the role should not contradict any of the types satisfied by that object unless the specification includes mechanisms to determine and resolve such inconsistencies, for example, by changing some of these types.

Roles may be created or deleted during the lifetime of a community. The role lifetime is contained within the community lifetime, and the period for which a particular enterprise object fulfils a given role is contained within the lifetime of that role.

The policies of a community apply to the behaviour of the objects in the community. Policies of the community may include prescriptions concerning the assignment of roles to enterprise objects.

NOTE –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 type of enterprise object.

Role is a specification concept; it is generally used as part of a type specification to link pieces of independent specification together consistently. When the community template containing the role is instantiated to create a composite enterprise 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 deleted, 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.

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.

7.7.2.1 Interface Roles

It is sometimes necessary to consider the interactions of a community as a composite enterprise object with its environment as the provision one or more services. This can be modelled by considering each service as an interface of the <c>-object. Then, in specifying the community it is necessary to establish the mapping between the interfaces (services) of the <c>-object and the roles of the enterprise objects in the community that are directly associated with the interactions that provide the service(s). These are interface roles. This mapping emphasises the fact that in the community specification these interactions must be modelled as being interactions involving enterprise objects of the community rather than as interactions of the <c>-object.

7.7.2.2 Community structure

Objects of a community may interact with objects outside of that community to achieve the objective of the community. If these interactions are identified by roles, then:

-- either the objects that interact must belong to some different community, in which case the roles that identify these interactions must be part of the specification of that different community,

-- or the objects outside of the that community interact with a composite object (referred to as a <c>-object) that is a composition of some of the objects of that community. In this case, the roles that identify these interactions and that are fulfilled by the objects of that community are interfaces of the <c>-object.

TEMPORARY NOTE – There is no consensus in the working group whether an enterprise specification can represent actions that are not in a behaviour identified by a role.

In the first case, this interaction is reduced to interaction of objects inside some community. The interactions between these objects are identified by the roles of this community.

In the second case, we consider the <c>-object as a member of a larger community. Then:

-- the <c>-object fulfils one or more roles in the larger community

-- the other objects of the larger community interact with the <c>-object through its interfaces.

The following are the structuring rules that prescribe the relation between the <c>-object and the larger community:

-- the interfaces of the <c>-object in the larger community correspond to the interface roles of the (smaller) community.

-- the roles that the <c>-object fulfils in the larger community identify only actions that are performed by objects of the core community

-- actions identified by other roles of the larger community that are interactions with the <c>-object are also identified by environment roles of the community.

TEMPORARY NOTE – It is not the intent of the working group that the term '<c>-object' appear in the Recommendation | International Standard. It is used here as a placeholder for an appropriate term.

NOTE – The structuring rules in Part 3 prescribe that an enterprise specification starts with a specification of a community that consists of the ODP system being specified and the objects in its environment. There are no objects in the specification outside this community.

7.7.3 Enterprise objects and actions

An enterprise object fulfils at least one role in at least one community and can be involved in actions in the following ways:

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

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

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

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

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

In the special case where an enterprise object is used in an action in which it also participates it is both an actor and a resource with respect to that action.

Editor's Note – This is problematical. A resource used in an action certainly participates in that action

When a resource is essential for some action, the action is constrained by the availability of that resource. Zero or more resources may be essential for an action.

Where a role in a community involves an enterprise object in actions only as an artefact, then it is an artefact role in that community.

Where a role in a community involves an enterprise object in any action as a resource, then it is a resource role in that community.

Where a role in a community involves an enterprise object 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, artefact roles and resource roles with respect to that community.

Editor's Note – This is (no longer) true.

NOTE – A role is specified in such a way that the behaviour associated with the role, the policies applying to the role, the responsibilities associated with the role, and the relationships with other roles are stated. For example, for each actor role there are complete descriptions of all actions of objects fulfilling that role, and for each action, identification of all the artefact roles mentioned in the action.

Editor's Note – 'Responsibility' is not a defined RM-ODP concept. 'Responsible' is used once in [3-8] and 'responsibility' is used several times in [3-13 et seq.].

'Complete' is ambiguous in this context.

7.7.4 Process rules

TEMPORARY NOTE – The following text may use more concepts than needed. The working group considers the choice of terms here a major issue. The working group invites National Body contributions proposing a minimal set of concepts (from action, activity, task, step, or others).

7.7.4.1 Specifying processes

A process is modelled as a directed acyclic graph of steps, in which each step:

-- is made possible by state changes of one or more enterprise objects, or the completion of one or more previous steps in the process;

-- results in state changes of one or more enterprise objects, some of which make possible the occurrence of subsequent steps in the same process.

NOTES –

1 – The use of 'acyclic' indicates that the trace, or history, of steps 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 step occurrences.

2 – 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?"

2 – Use of step instead of action here limits the use of the concept, process, to processes specified in the context of a community.

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

7.7.4.2 Relationships between process and other enterprise language concepts

Each action identified in a process is associated with at least one actor role. Different actions in a process can be associated with different actor roles. 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.

Editor's Note – Is the concept 'identified association of a result of an action with an interaction' clear?

Editor's Note – Is 'interaction between roles' shorthand for 'interaction of objects fulfilling roles'? Do we make this explicit? Should we? Would it be better to use 'interaction of objects fulfilling roles'?

Or is 'interaction between roles' shorthand for 'interaction in more than one behaviours identified by roles'? That is, an interaction that is in two (or more) behaviours because it is part of the behaviour of two or more objects: Role A includes an interaction of the object, anAobject, fulfilling that role with an object fulfilling Role B, aBobject (call this interaction 'A<->B'; Role B includes the same interaction, A<->B. Should we make this explicit? Would it be better to say what is meant here, rather than use a shorthand?

The difference between these two alternatives is that in the first it is not explicit that the interaction mentioned is in the behaviour identified by the role, though it may be necessary, depending on our theory of the use of roles. This is related to the question asked elsewhere whether there may be actions in an enterprise specification which are not in behaviours named by roles.

7.8 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.

Editor's note – This text uses the term 'principal' for a concept which is different from the 'principal' of 6.4.3.

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. Where a community is specified in the context of an implicit or explicit outer community; it cannot arbitrarily relax policies of that outer community or introduce policies that are in conflict with those of that outer community.

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 enterprise 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:

-- 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 subject to different constraints from those that applied before;

-- 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.

-- 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;

-- 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.

7.8.1 Scope of policies

Policies have varying scope. A policy may apply across a community, to a role or set of roles, or to a single action type.

NOTE – Examples: Across a community--all members of the community shall be paid-up subscribers. To a role--any enterprise object fulfilling actor role R may not also be a member of community C. To a single action type--invoices shall not be paid until at least 30 days after receipt.

The default policy maker and controller for an enterprise object is the owner of that object. Each enterprise object has at most one owner, even if that object is refined to or ownership is delegated to a set of cooperating objects. An enterprise object can be owned by itself. Object ownership becomes specified at instantiation, but can be transferred, delegated or relinquished (temporarily or permanently).

Editor's Notes:

1 - The previous text had "transferred or delegated (temporarily or permanently) or can be relinquished (permanently)." That fits with the meaning of 'to relinquish.'

2 - What enterprise object has ownership after some object (permanently) relinquishes ownership is not defined by this text. It may be intended that, if the owner does not relinquish ownership to another object, then the enterprise object has no owner.

3 - The text does not discuss specification of ownership at introduction.

NOTES:

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

2 – An enterprise specification should state whether and under what conditions all or some of the constraints of the roles of a community can be overridden by the owner.

7.8.2 Obligations, permissions, prohibitions, and authorisations

Policies that constrain behaviour are limiting statements about the choices that an enterprise object has in fulfilling an actor role.

TEMPORARY NOTES

1 – The following subclauses, 7.5.3.1 thru 7.5.3.4, may be overly prescriptive. The working group invites National Bodies to propose text that is less prescriptive.

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

Consider mention of collective behaviour.

2 – The working group invites National Bodies to propose a framework for policy languages.

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 enterprise objects.

Editor's Note - The editor has reordered the next three subclauses to follow the order of [2-11.2.4 to 2-11.2.6.]

7.8.2.1 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. Your editor has 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 are obligations where enabling conditions are always true, thus any interaction must conform to the obligation in addition to other constraints on that interaction.

7.8.2.2 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 or roles involved in that action

-- a predicate on behaviour

-- an authority which grants the permission

If a set of enterprise objects has this permission, then, when the predicate is true, that set of enterprise objects can take part in the action when filling the roles, by order of the authority.

Editor's Note - Does ‘can’ convey the meaning of permission? This may be more precise:

If a set of enterprise objects has this permission, then, when the predicate is true, there is no obligation for that set of enterprise objects not to take part in the action when filling the roles, by order of the authority.

7.8.2.3 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 or roles involved in that action

--a predicate on decisions influenced by policy

-- an authority which imposes the prohibition

If a set of enterprise objects has this prohibition, then, when the predicate is true, that set of enterprise objects cannot take part in the action when filling the roles, by order of the authority.

7.8.2.4 Authorisation

If an enterprise object has an authorisation to participate in an action, then other objects have no permission to deny that action to take place. If such a denial occurs, that is a violation of the authorisation and a failure.

7.8.3 Nesting of policy frameworks

TEMPORARY NOTE – The working group invites National Body contributions of an explicit description of a policy framework and rules for nesting such frameworks at an abstraction level suitable for this standard.

7.8.4 Enforcing policy

TEMPORARY NOTE – Most of this text may be a narrative and not normative text. The working group invites National Body contributions of appropriate normative text.

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.

7.8.5 Organisation of policy

TEMPORARY NOTE – The working group invites National Body contributions of 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) enterprise objects, to (collections of) enterprise 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 enterprise objects fulfilling roles in the community can participate.

The specification of enterprise behaviour 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 enterprise objects have the capability to perform particular sets of actions.

-- a delegation sub-model expressing the degree to and circumstances in which one enterprise 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.

7.8.5.1 Permission rules

Successful performance of an interaction between enterprise objects of a community requires that a set of permissions exists. Required permissions are either:

-- associated with a particular target role in that interaction, or

-- associated with that interaction as a whole.

TEMPORARY NOTE - The working group invites National Body comments on whether specification of permissions should be mandatory in an enterprise specification.

It is not necessary that there is an authorization to perform the interaction. If both authorization and permission for an interaction is missing, the interaction fails and therefore the enterprise object may fail in fulfilling its role.

Objects can pass permissions and authorizations between themselves. This passing is itself an interaction, and is subject to same permission rules.

7.8.5.2 Delegation rules

TEMPORARY NOTE – The working group invites National Body contributions of a small set of abstract structuring rules.

7.9 Enterprise object rules

An enterprise object is any object in an enterprise specification. The enterprise objects in a specification will be exactly the objects the specifier feels are necessary or desirable to specify the system from the enterprise viewpoint or to understand the enterprise specification.

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

An enterprise object may be refined as a community at a greater level of detail. 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.

Editor's Note - In "An enterprise object may be refined as a community at a greater level of detail" does not agree with one interpretation of the meaning and use of community: the interpretation used in the definition of <c>-object at subclause 6.1.6. In that interpretation, an enterprise object may be refined as a configuration of objects which objects are *some* of the objects of a community.

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

-- the community specification provides that the community includes the object,

-- the object is made part of the community at the time of community creation, or

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

Editor's Note - The editing instruction of the Paris meeting say to move this sentence from 7.5.1, Establishment of a community to 7.9, Enterprise object rules. The editor feels this text belongs somewhere in 7.5 Lifecycle of a community, and invites National Body comment.

7.10 Force rules

TEMPORARY NOTE – These are a first attempt at rules for the use of the force concepts. The working group invites National Body comment on and suggested improvements for this text as well as additional rules.

The working group uses the term, 'force,' in this sense: the real import or significance (of a document, statement, or the like). [Oxford English Dictionary, sense 9] The use of these concepts is to specify the import or significance of actions in an enterprise specification. The working group welcomes National Body comment on the term and suggestions for a different term.

The working group invites National Body comment and suggested text relating these concepts to the policy concepts and delegation concepts in Clause 7.

The environment of an ODP system includes natural persons or other entities considered to have some of the rights, powers and duties of a natural person; these entities are modeled by parties. The environment also may include other information processing systems in addition to that ODP system. This Reference Model provides rules for the specification of an ODP system in terms of the force that actions of that system have in the universe of entities modeled by parties in its environment.

An enterprise specification describes the authority delegated to an ODP system in terms of:

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

--the authority that each party has delegated;

--the duration and conditions of the delegation;

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

By each such delegation, the system becomes an agent of the party delegating, and the party becomes a principal of the system.

Actions of an ODP system that are an exercise of delegated authority are acts. Acts cause states of affairs in that part of the universe of discourse modeled by the parties in the environment of that system. The force of an act specifies which kinds of states of affairs the act causes.

7.10.1 Authority rules

For each authority delegated, an enterprise specification describes the force of actions of the system in exercising that authority, including authority:

--to make commitments, these bind the principal

--to issue declarations, these create a state of affairs just as if the principal had made the declaration;

--to delegate an authority to a component of the system, to another system or to a party, these cause that agent to have the authority;

--to communicate descriptions behalf of the principal;

--to make evaluations on behalf of the principal;

--to give instructions to a person or machine on behalf of the principal;

--to make a prescription which establishes a rule with the same force as if the principal had made the prescription.

TEMPORARY NOTE – We are short of terms here. This should probably not be ‘the principal’ (an enterprise object), but ‘the entity modelled by the principal.’ Lacking a succinct term for that concept, it will be necessary to correct this clause by spelling it out every time.

7.10.2 Delegation rules

When an ODP system is an agent with authority to delegate, and it may (subject to the provisions of that authority) delegate (authority, a function, …) to a party or other system. That party or other system shall then be an agent of the principal and have the same authority as if delegated directly by the principal.

In general, a principal is responsible for the acts of its agents.

8 Compliance

An enterprise specification complilant with this Recommendation | International Standard shall use the concepts defined in clause 6 and those in clause 5.1 of ITU-T Rec. X.903 | ISO/IEC 10746-3, 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 explanations of the relationships between the concepts concerned and those defined in clause 6.

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

9 Conformance and reference points

Editor's Note – A clause on conformance and reference points in enterprise specifications is required for this Recommendation | International Standard. The working group invites National Body contributions. [See 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.

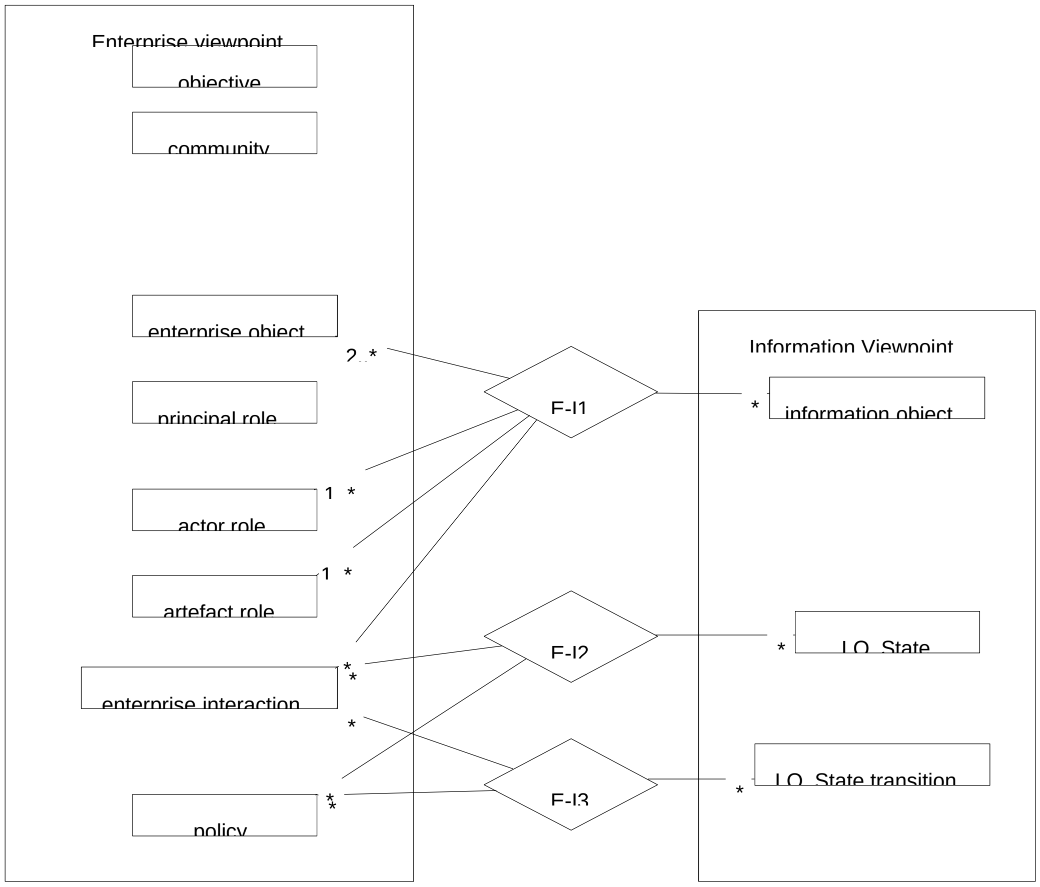

Figure 10- 1 Correspondences between the enterprise viewpoint and the information viewpoint

Editor's Notes:

1 – From comment 0014 TL of the Japan National Body, accepted by the working group: “Roles defined in 6.4 (and possibly 6.2) should be added in the figures.

“If modification of the figures is difficult, it should be noted, at least, that there may be other roles than ones described in the figures and that mapping of these roles into other viewpoints may depend on details of each ODP system.”

2 – From comment 26 TH of the Finland National Body, accepted by the working group: “The correspondence rules between enterprise language and other ODP viewpoint languages are not defined in terms of ODP vocabulary.

“In the illustrations, for enterprise viewpoint, the following boxes should be used:

-- Community

-- Objective

-- Scoping statement

-- Role

-- Process

-- Task

-- Actor role

-- Artefact role

-- Policy"

"For information viewpoint, the following concepts/boxes should be used:

-- Information object

-- Dynamic schema

-- Static schema

-- Invariant schema"

Your editor does not have the tools to modify the figure.

The correspondences between enterprise viewpoint concepts and information viewpoint concepts include:

-- One or more information objects can represent the information content of an enterprise object which fulfils any kind of role in the enterprise specification.

-- Invariant, static and dynamic schemata of information objects are governed by the enterprise viewpoint policies. The policies each information object must obey are those related to the whole community, those related to the enterprise object involved, and activities of that enterprise object as far as the information object is participating the activity.

-- A dynamic schema is associated with a task.

10.2 Enterprise and computational specification correspondences

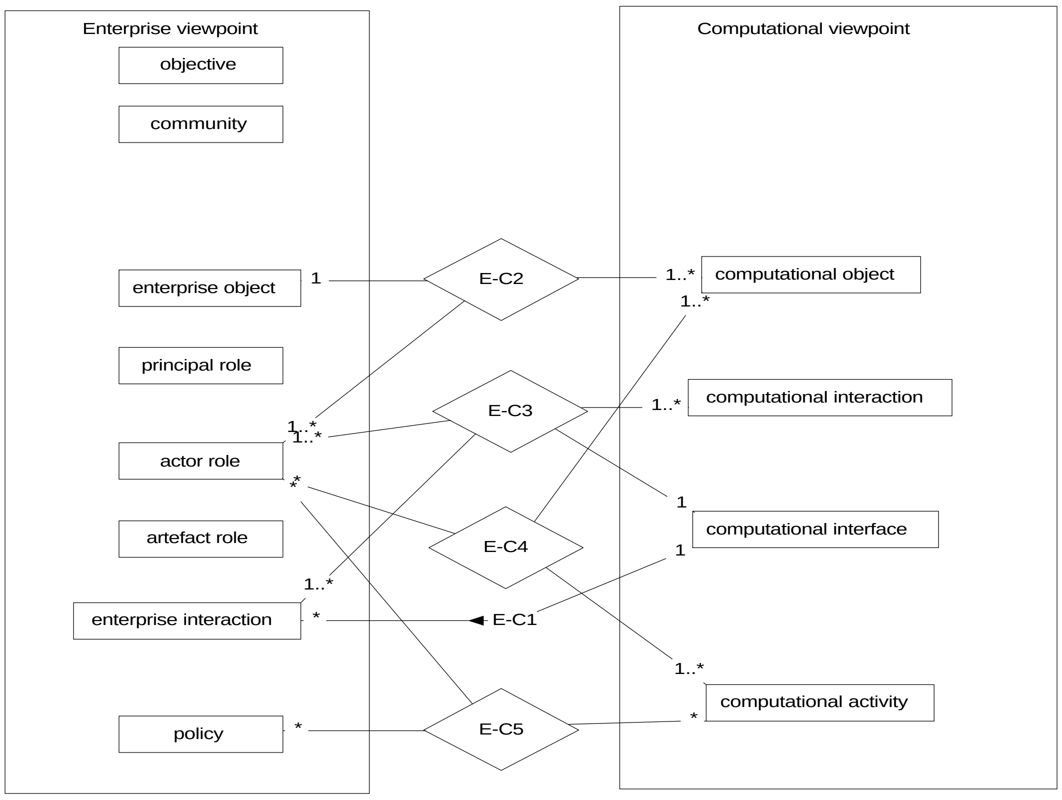

Figure 10- 2 Correspondences between the enterprise viewpoint and the computational viewpoint

Editor's Notes:

1 – From comment 0014 TL of the Japan National Body, accepted by the working group:

1 – There is no coincidence between "E-CX" in the figure and its explanation. Numbering in the figure should be "E-C1", "E-C2", "E-C3", "E-C4" and "E-C5" from the top to the bottom.

2 – From comment 26 TH of the Finland National Body, accepted by the working group: “For computational viewpoint, the following concepts/boxes should be used:

-- Computational object behaviour

-- Computational interaction

-- Computational interface

-- Binding object”

The correspondences between enterprise viewpoint concepts and computational viewpoint concepts include

-- A computational object may include multiple computational interfaces each of which represent a computational object behaviour. One or more computational interfaces can represent the computations related to an enterprise object.

-- One or more interactions of a computational object are related to a task.

-- Computational object behaviour is restricted by policies.

-- Binding object is related to cobehaviour of enterprise objects.

-- A computational interface should conform to the type of an enterprise interaction.

10.3 Enterprise and engineering specification correspondences

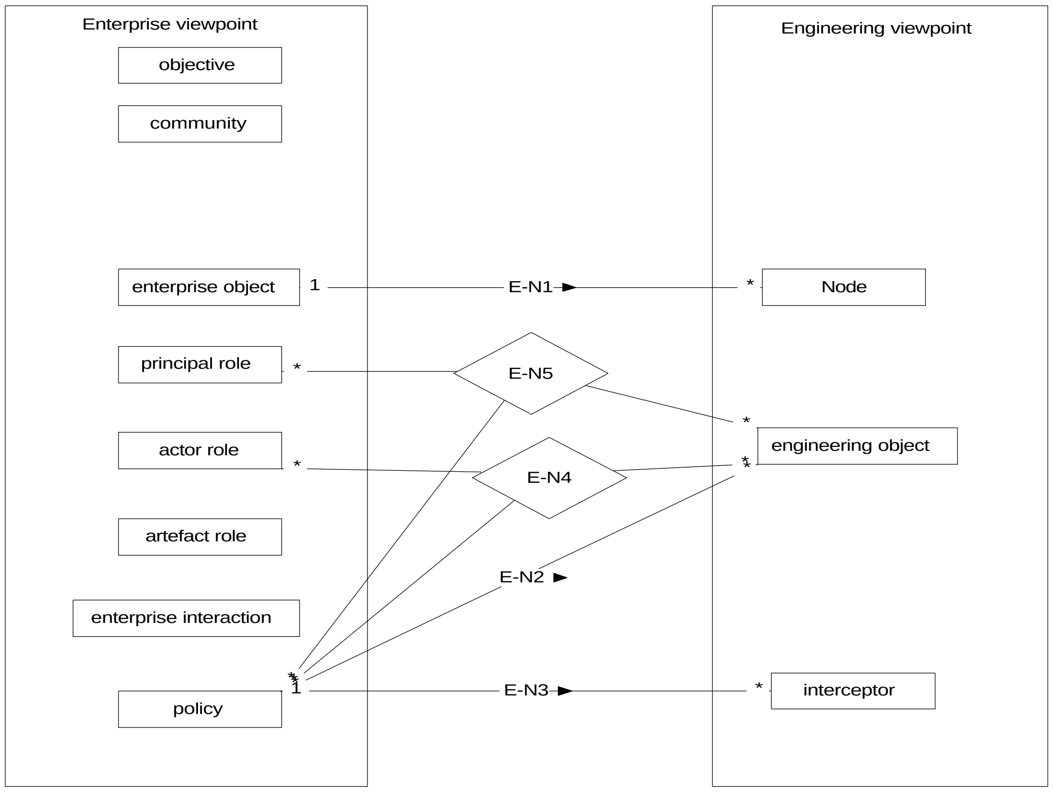

Figure 10- 3 Correspondences between the enterprise viewpoint and the engineering viewpoint

Editor's Notes:

1 – From comment 0016 TL of the Japan National Body, accepted by the working group: “The important staffs in Engineering viewpoint such as "stub", "binder" and "protocol" should be added.

“-- Draw a line with a label "E-N6" from "enterprise interaction" and "policy" to "stub", "binder" and "protocol".

2 – From comment 26 TH of the Finland National Body, accepted by the working group: “For engineering viewpoint, the following concepts/boxes should be used:

-- Basic engineering object

-- Node

-- Interceptor

The correspondences between enterprise viewpoint concepts and engineering viewpoint concepts include

-- One or more basic engineering objects may represent an enterprise object.

-- An enterprise object may locate at one or more nodes.

-- A domain is a specific community type in the enterprise language. One or more interceptors may be needed to implement a domain boundary.

--An enterprise interaction with a policy should provide some constraints for behaviours of an engineering interaction based on "stub", "binder" and "protocol".”

Annex A Overall structure of an enterprise specification

TEMPORARY NOTE – This no longer matches clause 7. The working group invites National Body contributions.

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.

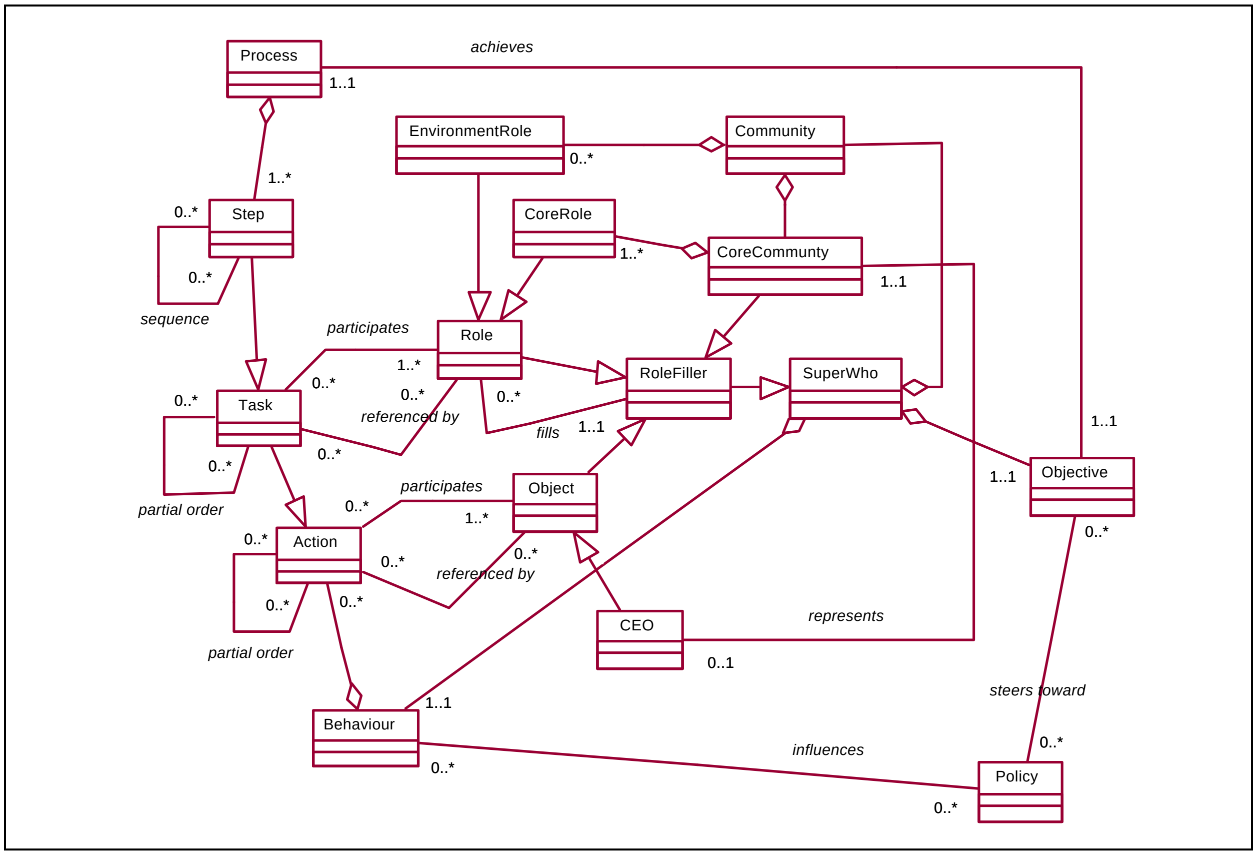

Figure A-1 Working language model

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>

NOTE - 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>

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

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)

Figure A-3 Extended BNF Notation

Index