

ISO/IEC JTC 1/SC 32 N 0530

Date: 2000-09-18

REPLACES: --

<p style="text-align: center;">ISO/IEC JTC 1/SC 32</p> <p style="text-align: center;">Data Management and Interchange</p> <p style="text-align: center;">Secretariat: United States of America (ANSI) Administered by Pacific Northwest National Laboratory on behalf of ANSI</p>

DOCUMENT TYPE	Liaison Organization Contribution
TITLE	FCD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint
SOURCE	JTC1/SC7 Secretariat
PROJECT NUMBER	
STATUS	SC32 members can send comments to Jean Bérubé (Liaison to SC 7) jberube@attglobal.net . It is of interest to ex-ODP and CSMF members
REFERENCES	
ACTION ID.	COM
REQUESTED ACTION	
DUE DATE	
Number of Pages	43
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile: +1 703 671 9180; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite <http://www.jtc1sc32.org/>

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI



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

ISO/IEC JTC1/SC7 N2359

2000-09-09

Document Type	FCD Ballot
Title	FCD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint
Source	JTC1/SC7 Secretariat
Project	07.77
Status	FCD Ballot
References	N2358
Action ID	FYI or ACT
Due Date	2001-01-09
Mailing Date	2000-09-09
Distribution	SC7_AG
Medium	Encoded Acrobat
No. of Pages	41
Note	Joint project with ITU-T (ITU-T X911).



ISO/IEC JTC1/SC7 FCD 15414	
Date 2000-09-09	Reference number ISO/JTC 1/SC 7 N2359
Supersedes document N2252	

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) 2001-01-09 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: FCD 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: 39



Vote on FCD 15414	
Date of circulation 2000-09-09	Reference number ISO/JTC 1/SC 7 N2359
Closing date 2001-01-09	

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: FCD 15414: Information Technology - Open Distributed Processing - Reference Model - Enterprise Viewpoint

Project: 07.77

Vote:

- APPROVAL OF THE DRAFT AS PRESENTED
- APPROVAL OF THE DRAFT WITH COMMENTS AS GIVEN ON THE ATTACHED
 - general:
 - technical:
 - editorial:
- DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED
 - Acceptance of these reasons and appropriate changes in the text will change our vote to approval
- ABSTENTION (FOR REASONS BELOW):

P-member voting:
National Body (Acronym)

Date:
YYYY-MM-DD

Submitted by:
Your Name



ITU-T X.911 ISO/IEC 15414

Date: 00-07-10

**Committee Draft
Madrid 2000 Output**

10 July 2000

**ISO/IEC JTC 1/SC 7
Software Engineering
Secretariat: Canada (SCC)**

Doc Type: Final 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 final committee draft; output from May-June 2000 Madrid meeting

Action: For FCD ballot

Distribution: SC 7

Medium: E

Number of Pages: 38

Version: 303

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

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.911

(Final Committee Draft 7/00)

COMMON TEXT DRAFT

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

ITU-T Recommendation X.911

Version 303

CONTENTS

	<i>Page</i>
0	Introduction..... 1
0.1	RM-ODP..... 1
0.2	This Recommendation International Standard..... 1
1	Scope..... 3
2	Normative references 3
	Identical ITU-T Recommendations International Standards 3
3	Definitions..... 4
3.1	Definitions from ODP standards 4
3.1.1	Modelling concept definitions 4
3.1.2	Viewpoint language definitions..... 5
3.2	Definitions from ODP standards refined or extended in this standard..... 5
4	Abbreviations 5
5	Overview and motivation 5
6	Concepts..... 6
6.1	General concepts..... 7
6.2	Role concepts 8
6.3	Policy concepts..... 9
6.4	Force concepts 9
7	Structuring Rules..... 10
7.1	Overall structure of an enterprise specification..... 10
7.2	Contents of an enterprise specification 11
7.3	Community rules..... 11
7.3.1	Specification of a community..... 11
7.3.2	Relationships between communities 12
7.4	Enterprise object rules..... 13
7.5	Common community types..... 13
7.5.1	Domain community type..... 14
7.5.2	Federation community type..... 14
7.5.3	Ownership Community Type 14
7.6	Lifecycle of a community 14
7.6.1	Establishing a community..... 14
7.6.2	Changes in a community..... 14
7.6.3	Terminating a community..... 15
7.7	Objective rules 15
7.8	Behaviour rules..... 15
7.8.1	Roles and processes..... 15
7.8.2	Role rules..... 16
7.8.3	Interface roles and interactions between communities 17
7.8.4	Enterprise objects and actions 17
7.8.5	Process rules 18
7.9	Policy rules 19
7.9.1	General policy rules 19
7.9.2	Obligations, permissions and prohibitions..... 19
7.9.3	Nesting of policy frameworks..... 20
7.9.4	Policy violations..... 21
7.9.5	Organisation of policy..... 21
7.10	Force rules 22
7.10.1	Delegation rules 22
7.10.2	Authority rules 22

1		7.10.3 Commitment rules.....	23
2		7.10.4 Declaration rules.....	23
3		7.10.5 Prescription rules.....	23
4	8	Consistency rules.....	23
5		8.1 Viewpoint correspondences.....	23
6		8.2 Enterprise and information specification correspondences.....	24
7		8.2.1 Concepts related by correspondences.....	24
8		8.2.2 Required correspondences.....	24
9		8.2.3 Required correspondence statements.....	25
10		8.3 Enterprise and computational specification correspondences.....	25
11		8.3.1 Concepts related by correspondences.....	25
12		8.3.2 Required correspondences.....	25
13		8.3.3 Required correspondence statements.....	25
14		8.4 Enterprise and engineering specification correspondences.....	26
15		8.4.1 Concepts related by correspondences.....	26
16		8.4.2 Required correspondences.....	26
17		8.4.3 Required correspondence statements.....	26
18	9	Relations between standards and product development.....	26
19		9.1 Compliance, conformance, testing and consistency.....	27
20		9.2 Completeness.....	27
21		9.3 Scoping statement.....	27
22	10	Enterprise Language Compliance.....	28
23	11	Conformance and reference points.....	28
24	Annex A	Overall structure of an enterprise specification.....	30
25	Index		32

1 **Foreword**

2 This is the initial final committee draft of ITU | ISO/IEC Common Text for Recommendation X.911 | International
3 Standard 15414: Information Technology—Open Distributed Processing—Reference Model—Enterprise Language. It
4 is the draft output of the May-June 2000 Madrid meeting.

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

6 The work on this draft Recommendation | International Standard is done by ISO/IEC JTC 1/SC 7/WG 17, originally
7 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.

8

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

11

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

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

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

19 **Indexing** The project editor requests suggestions for indexing this document.

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

21 Changes to ITU template; these must be changed back:

22 - Addition of style, Editors Note

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

24 - Font

25

26

27

28

1 **0 Introduction**

2 The rapid growth of distributed processing has led to the adoption of the Reference Model of Open Distributed
3 Processing (RM-ODP). This Reference Model provides a co-ordinating framework for the standardisation of open
4 distributed processing (ODP). It creates an architecture within which support of distribution, interworking, and
5 portability can be integrated. This architecture provides a framework for the specification of ODP systems.

6 The Reference Model of Open Distributed is based on precise concepts derived from current distributed processing
7 developments and, as far as possible, on the use of formal description techniques for specification of the architecture.

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

10 **0.1 RM-ODP**

11 The RM-ODP consists of:

12 - ITU-T Recommendation X.901 | ISO/IEC 10746-1: **Overview**: which contains a motivational overview of ODP,
13 giving scoping, justification and explanation of key concepts, and an outline of the ODP architecture. It contains
14 explanatory material on how the RM-ODP is to be interpreted and applied by its users, who may include standards
15 writers and architects of ODP systems. It also contains a categorisation of required areas of standardisation expressed
16 in terms of the reference points for conformance identified in ITU-T Recommendation X.903 | ISO/IEC 10746-3. This
17 part is not normative.

18 - ITU-T Recommendation X.902 | ISO/IEC 10746-2: **Foundations** which contains the definition of the concepts and
19 analytical framework for normalised description of (arbitrary) distributed processing systems. It introduces the
20 principles of conformance to ODP standards and the way in which they are applied. This is only to a level of detail
21 sufficient to support ITU-T Recommendation X.903 | ISO/IEC 10746-3 and to establish requirements for new
22 specification techniques. This part is normative.

23 - ITU-T Recommendation X.903 | ISO/IEC 10746-3: **Architecture**: which contains the specification of the required
24 characteristics that qualify distributed processing as open. These are the constraints to which ODP standards must
25 conform. It uses the descriptive techniques from ITU-T Recommendation X.902 | ISO/IEC 10746-2. This part is
26 normative.

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

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

33 **0.2 This Recommendation | International Standard**

34 ITU-T Recommendation X.903 | ISO/IEC 10746-3 defines a framework for the specification of ODP systems
35 comprising

36 1) five viewpoints, called enterprise, information, computational, engineering and technology, which provide a basis for
37 the specification of ODP systems;

38 2) a viewpoint language for each viewpoint, defining concepts and rules for specifying ODP systems from the
39 corresponding viewpoint.

40 The purpose of this Recommendation | International Standard is to:

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

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

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

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

5 This Recommendation | International Standard contains, for the convenience of the reader, some text taken verbatim
6 from clauses 5 and 10 of ITU-T Recommendation X.903 | ISO/IEC 10746-3: Part 3: Architecture. Such text is marked
7 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
8 discrepancies in these cases, the text of ITU-T Recommendation X.903 | ISO/IEC 10746-3 is authoritative.

9 This Recommendation | International Standard also contains some text which is a modification of text ITU-T
10 Recommendation X.903 | ISO/IEC 10746-3: Part 3: Architecture. Such text is marked by a reference like this: [see also
11 3-5.9]. The modifications are authoritative with respect to the enterprise language.

12 This Recommendation | International Standard contains these annexes:

13 Annex A: Overall structure of an enterprise specification

14 This annex is not normative.

15

16

1 **INTERNATIONAL STANDARD**

2
3
4 **ITU-T RECOMMENDATION**

5 **INFORMATION TECHNOLOGY—OPEN DISTRIBUTED PROCESSING—**
6 **REFERENCE MODEL—ENTERPRISE LANGUAGE**

7 **1 Scope**

8 This Recommendation | International Standard provides:

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

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

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

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

20 This standard is intended for use in preparing enterprise viewpoint specifications of ODP systems, and in
21 developing notations and tools to support such specifications.

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

24 **2 Normative references**

25 The following ITU-T Recommendations | International Standards contain provisions which, through reference in
26 this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the
27 editions indicated were valid. All Recommendations | International Standards are subject to revision, and parties
28 to agreements based on this Recommendation | International Standard are encouraged to investigate the
29 possibility of applying the most recent editions of the Recommendations | International Standards listed below.
30 Members of IEC and ISO maintain registers of currently valid International Standards. The ITU-T Secretariat
31 maintains a list of the currently valid ITU-T Recommendations.

32 **Identical ITU-T Recommendations | International Standards**

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

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

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

3 3 Definitions

4 3.1 Definitions from ODP standards

5 3.1.1 Modelling concept definitions

6 This Recommendation | International Standard makes use of the following terms as defined in ITU-T
7 Recommendation X.902 | ISO/IEC 10746-2

- | | |
|--|-------------------------------------|
| 8 -- action; | 32 -- object; |
| 9 -- behaviour (of an object) ; | 33 -- obligation; |
| 10 -- composite object; | 34 -- ODP standards; |
| 11 -- composition; | 35 -- ODP system; |
| 12 -- configuration (of objects) ; | 36 -- perceptual reference point; |
| 13 -- conformance; | 37 -- permission; |
| 14 -- conformance point; | 38 -- policy; |
| 15 -- contract; | 39 -- postcondition; |
| 16 -- <X> domain; | 40 -- precondition; |
| 17 -- entity; | 41 -- programmatic reference point; |
| 18 -- environment contract; | 42 -- prohibition; |
| 19 -- environment (of an object) ; | 43 -- proposition; |
| 20 -- epoch; | 44 -- quality of service; |
| 21 -- establishing behaviour; | 45 -- reference point; |
| 22 -- incremental modification; | 46 -- refinement; |
| 23 -- instance (of a type) ; | 47 -- responding object; |
| 24 -- instantiation (of an <X> template) ; | 48 -- role; |
| 25 -- interface; | 49 -- state (of an object) ; |
| 26 -- internal action; | 50 -- subtype; |
| 27 -- interworking reference point; | 51 -- system; |
| 28 -- invariant; | 52 -- <X> template; |
| 29 -- liaison; | 53 -- terminating behaviour; |
| 30 -- location in space; | 54 -- type (of an <X>); |
| 31 -- location in time; | 55 -- viewpoint (on a system). |

1 3.1.2 Viewpoint language definitions

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

- | | |
|-------------------------------|------------------------------|
| 4 -- community; | 11 -- information viewpoint; |
| 5 -- computational interface; | 12 -- invariant schema; |
| 6 -- computational viewpoint; | 13 -- <viewpoint> language; |
| 7 -- dynamic schema; | 14 -- static schema; |
| 8 -- engineering viewpoint; | 15 -- technology viewpoint. |
| 9 -- enterprise viewpoint; | 16 . |
| 10 -- <X> federation | |

17 3.2 Definitions from ODP standards refined or extended in this standard

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

- 20 -- policy

21 4 Abbreviations

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

25 5 Overview and motivation

26 TEMPORARY NOTE – Canada has made this comment: “It is not clear if the target audience of this document is.
27 This can affect the style of the document.

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

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

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

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

40 Editor’s Note – The use of the term, ‘Recommendation | International Standard,’ at this point
41 in the document contravenes Clause 1 of Rules for presentation of ITU | ISO/IEC Common
42 Text, which prescribes that “a term which is descriptive of the nature of the common text
43 should be used when the document refers to itself” in clauses after the Scope clause.

1 Accordingly, a descriptive term for this document must be chosen. Possible terms include
2 ‘reference model,’ ‘prescription of the ODP enterprise language’ and ‘language.’ (This
3 Recommendation | International Standard is: Information Technology—Open Distributed
4 Processing—Reference Model—Enterprise Language.

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

12 The enterprise language provides the terms and structuring rules to specify the purpose, scope and policies for
13 an ODP system in a manner that is meaningful for the stakeholders for that system, including the owners, the
14 users and the developers. An enterprise specification describes the behaviour of the system within the
15 environment with which it interacts. Such an environment can be a technical environment (e.g., the software
16 and hardware environment of a service component) or a social or business organisation (e.g., a group of co-
17 operating companies, a particular service inside a company).

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

20 An enterprise specification of an ODP system is an abstraction of the system and a larger environment in
21 which the ODP system exists, describing those aspects that are relevant to specifying what the system is
22 expected to do in the context of purpose, scope and policies of its environment (technical, organisational). It
23 describes the behaviour assumed by those who interact with the ODP system. It explicitly includes those
24 aspects of the environment that influence the behaviour of the ODP system – environmental constraints are
25 captured as well as usage and management rules.

26 An important objective of an enterprise specification is to support an agreement (for example, as part of the
27 contract for the supply of a system) between the potential clients of an ODP system and the provider of that
28 system. Both parties should be able to write, read and discuss such a specification, the clients to be sure of the
29 expected behaviour of the system that they will get, and the provider to be clear about the behaviour to be
30 realised by the system being provided. Thus, two types of presentation of the enterprise specification may need
31 to be considered for the same system. One presentation may need to provide a view of the specification in
32 terms that are understood by the clients. A second presentation may be needed to present the specification in
33 terms that more directly relate to its realisation. Both types of presentation address enterprise considerations as
34 they concern the system.

35 The motivation for a standard enterprise language is to support standardized techniques for specification. This
36 improves communication and helps create consistent overall specifications. One way of using the enterprise
37 viewpoint is to discover enterprise objects by focusing on purpose, scope and policies of the system, and then
38 define other viewpoint models, e.g., computational objects, based on the correspondence with enterprise
39 objects.

40 **6 Concepts**

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

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

44 --the concepts defined in this clause.

1 The concepts defined in this clause include both new concepts and refinements of concepts from ITU-T X.902
2 | ISO/IEC 10746-2 and ITU-T X.903 | ISO/IEC 10746-3. The grouping into subclauses and the headings of
3 the subclauses of this clause are not normative.

4 **6.1 General concepts**

5 **6.1.1 Purpose (of a system):** The practical advantage or intended effect of the system.

6 **6.1.2 Objective:** Statements of preference about possible future states, which influence the choices within
7 some behaviour.

8 NOTES

9 1 – Some objectives are ongoing, others are achieved once met.

10 2 – The choices influenced by an objective are not necessarily choices of the holder of the objective.

11 Editor's Note - Could it be that an objective is not a statement? That it is a statement of an
12 objective that is a statement? If so, do we want: Objective: Preferences about possible future
13 states which influence the choices within some behaviour.

14 **6.1.3 Scope (of a system):** The behaviour that system is expected to exhibit.

15 TEMPORARY NOTE – The working group invites National Body consideration whether there is a need for a concept
16 that expresses what the system is capable of doing. (Such a concept would distinguish the delivered behaviour from
17 the expected behaviour.)

18 **6.1.4 Purposeful selection:** An action selecting one of a set of possible behaviours where the behaviour
19 initiated is the one that best suits an objective, as determined by some selection criteria.

20 NOTES:

21 1 – A purposeful selection may take place as an internal action of an enterprise object or by the collective behaviour of
22 several enterprise objects.

23 2 – The set of possible behaviours may not be predetermined and the selection criteria may not be explicit. This may,
24 in particular, be the case when the action of selection involves parties.

25 **6.1.5 Process:** A collection of steps taking place in a prescribed manner and leading to the accomplishment of
26 some result.

27 NOTES

28 1 – A process may have multiple starting points.

29 2 – The activity structure concepts provided in subclause 13.1 of ITU-T Recommendation X.902 | ISO/IEC 10746-2
30 may be used, after substitution of 'step' for 'action' and 'process' for 'activity,' to specify the structure of a process.

31 3 – An enterprise specification may define types of processes and may define process templates.

32 **6.1.6 Step:** An abstraction of an action, used in a process, that may hide objects with which that action is
33 associated.

34 **6.1.7 S-community:** A community in which an ODP system is represented as a single enterprise object
35 interacting with its environment

36 **6.1.8 C-object:** A composite enterprise object that represents a community. Components of a c-object are
37 objects of the community represented.

38 **6.1.9 Party:** An enterprise object modelling a natural person or any other entity considered to have some of
39 the rights, powers and duties of a natural person.

40 NOTES

41 1 – Examples of parties include enterprise objects representing natural persons, legal entities, governments and their
42 parts, and other associations or groups of natural persons.

43 2 – Parties are responsible for their actions and the actions of their agents.

44 **6.1.10 Machine:** An enterprise object modelling an automated system.

1 NOTE – Examples of machines include computers and the software they support, systems of computers, other
2 electronic devices and mechanical devices.

3 TEMPORARY NOTE - The working group invites National Body comment on the need for this concept.

4 **6.1.11 Owner (of an <X>):** The party or one of several parties having the right to control the use and disposal
5 of the <x>.

6 NOTES

7 1 – Commonly, this is a party paying for the specification, construction, instantiation, or current operation of the <x>.
8 This party will typically grant authorisation to use the <x> to other parties or their agents.

9 2 – Ownership is restricted to parties to enable assignment of responsibility for actions.

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

14 **6.2 Role concepts**

15 **6.2.1 Actor (with respect to an action):** An enterprise object that participates in the action.

16 NOTE - It may be of interest to specify which of the actors involved initiates the action.

17 **6.2.2 Artefact (with respect to an action):** An enterprise object that is referenced in the action.

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

19 **6.2.3 Resource:** An artefact which is essential to some behaviour and which requires allocation or may
20 become unavailable because it is in use or used up.

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

22 **6.2.4 Actor role (with respect to a community):** A role in that community in which the enterprise object
23 filling the role is involved in at least one action of the role as an actor.

24 **6.2.5 Artefact role (with respect to a community):** A role in that community in which the enterprise object
25 filling the role is involved in all actions of the role only as an artefact.

26 **6.2.6 Resource role (with respect to a community):** A role in that community in which the enterprise object
27 filling the role is involved in any actions of the role only as a resource.

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

29 **6.2.7 Interface role (with respect to a community):** A role of a community that identifies actions that can be
30 performed by a c-object representing that community.

31 Editor's Note - Should this be interactions?

32 **6.2.8 Agent:** An enterprise object that has been delegated (authority, responsibility, a function, etc.) by and
33 acts for another enterprise object (in exercising the authority, carrying out the responsibility, performing the
34 function, etc.).

35 NOTES –

36 1 – An agent may be a party or may be the ODP system or one of its components. Another system in the environment
37 of the ODP system may also be an agent.

38 2 – The delegation may have been direct, by a party, or indirect, by an agent of the party having authorisation from the
39 party to so delegate.

40 TEMPORARY NOTE – This term is intended to have a meaning that follows one standard meaning of 'agent,' that in
41 the pair agent/principal. This meaning may be different from a currently popular use of 'agent' in computer software
42 circles, which may be closer to that in the pair agent/patient. In Madrid, the working group decided not to change this
43 term.

44 **6.2.9 Principal:** A party that has delegated (authority, a function, etc.) to another.

1 **6.2.10 Contracting party (with respect to a contract):** A party that agrees to that contract.

2 **6.3 Policy concepts**

3 **6.3.1 Policy:** A set of rules related to a particular purpose. A rule can be expressed as an obligation, an
4 authorisation, a permission or a prohibition.

5 NOTES:

6 1 – Not every policy is a constraint. Some policies represent an empowerment.

7 2 – This definition refines 2-11.2.7.

8 **6.3.2 Authorisation:** A prescription that a particular behaviour must not be prevented.

9 NOTE – Unlike a permission, an authorisation is an empowerment

10 **6.3.3 Violation:** An action contrary to a rule.

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

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

13 **6.4 Force concepts**

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

16 **6.4.1 Act:** A behaviour of parties or agents initiated by a purposeful selection.

17 NOTES:

18 1 - An act is therefore intentional, chosen to suit an objective.

19 2 - The enterprise object or objects participating in an act may be parties or machines acting as agents.

20 **6.4.2 Commitment:** An act resulting in an obligation by one or more of the participants in the act to comply
21 with a rule or perform a contract.

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

24 **6.4.3 Declaration:** An act that establishes the truth of a sentence referring to the environment of the object
25 making the declaration.

26 NOTE – The essence of a declaration is that, by virtue of the act of declaration itself, it causes a state of affairs to
27 come into existence outside the object making the declaration.

28 **6.4.4 Delegation:** The action of assigning authority, responsibility or a function to another object.

29 NOTES:

30 1 - Delegation may be from a party to a machine.

31 2 - Delegation may assign responsibility for making purposeful selections. In this case the action of delegation
32 provides the objective and may provide the selection criteria.

33 Editor's Note - We are leaving unspoken the other possibility raised by RM-ODP: transfer, as
34 opposed to delegation. [3-5.2]

35 **6.4.5 Evaluation:** An act that assigns a value to something.

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

38 **6.4.6 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 areas of interest of the specification users:

- roles fulfilled 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 may be represented as a single enterprise object interacting with its environment. This is referred to as the s-community. Whether the specification actually includes that level of abstraction is left for the specifier to decide.

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

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 are inconsistent with the objective 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 is defined in terms of its intended behaviour and in enterprise language this is expressed in terms of roles, processes, policies and their relationships.

A complete ODP system specification indicates rules for internal consistency in terms of relationships between various viewpoint specifications. Furthermore, a complete enterprise specification contains conformance rules that define the required behaviour of the described ODP system, thus separating out the parts of the specification included for the purposes of analysis, understandability and communicability of the specification.

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.

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.

1 **7.2 Contents of an enterprise specification**

2 An enterprise specification is composed of specifications of the elements explained in clause 7.1 (communities,
3 objects, roles, behaviour, ...).

4 Depending on the specifier's choice and desired level of detail, each of these elements can be specified by

5 -- the characteristics of the element, or

6 -- the type or types of the element, or

7 -- a template of the element.

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

10 NOTES:

11 1 – No recommendations are made about the relative merits of modelling from top-down or bottom-up. Nor is there a
12 recommended sequencing of the development of viewpoint specifications.

13 2 – It is a design choice whether a specification deals with a specific implementation by, for example, identifying
14 individual enterprise objects, or deals with a more flexible architecture by identifying types and assignment rules for
15 accepting enterprise objects.

16 Editor's Note - Does 'assignment rules for accepting enterprise objects' mean rules for
17 assigning enterprise object to roles?

18 **7.3 Community rules**

19 **7.3.1 Specification of a community**

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

23 -- states the objective for which the community exists,

24 -- governs the structure, the behaviour and the policies of the community,

25 -- constrains the behaviour of the members of the community,

26 -- states the assignment of enterprise objects to roles.

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

29 NOTES

30 1 – The concept, contract, is defined in ITU-T X.902 | ISO/IEC 10746-2.

31 2 – There is no requirement that the parties forming the contract fulfil roles in the community. Indeed, the lifetime of a
32 community can extend beyond the lifetime of the parties to the community contract.

33 Editor's Note – Note 2 (no parties to contract need fulfill roles) might be read as
34 contradicting the normative text (which suggests that at least some of the enterprise objects
35 of the community ("the enterprise objects") form the contract, if it is formed at the time of
36 community establishment.

37 The behaviour of the community expresses how it meets its objective. In the context of the community, the
38 objectives of members of the community are constrained to conform to its objective.

39 The behaviour of the community is defined in terms of one or more of the following elements:

40 -- the roles of the community (including those roles which define how a community interacts with
41 its environment),

42 -- the processes that take place in the community,

- 1 -- the way the roles and processes are combined and,
- 2 -- policies that apply to the roles and processes.

3 A behaviour of a community is a composition of behaviours that are identified by the roles of the community
4 or a composition of steps in the processes that take place in that community. Constraints on these behaviours
5 must be consistent with the constraints specified by the roles of the community, by the processes and by the
6 relationships between these roles.

7 Editor's Note - The project editor, after redrafting the first sentence of this paragraph
8 according to the instructions in the disposition of comments, feels that the current draft is
9 unclear on the relationship of roles and processes. National Bodies may wish to provide
10 additional text explaining this relationship.

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

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

- 14 -- roles;
- 15 -- policies for assignment of enterprise objects to roles;
- 16 -- relationships between roles;
- 17 -- relationships of roles to processes;
- 18 -- policies that apply to roles and to relationships between roles;
- 19 -- policies that apply to relationships between enterprise objects in the community;
- 20 -- behaviour that changes the structure or the members of the community during the lifetime of
21 that community.

22 NOTES

- 23 1 – Types of communities or a community template may be used in the specification of a community.
- 24 2 – Types of communities may be related by refinement.
- 25 3 – A family of related contracts may be generated from a contract template. Some aspects of the contract (e.g.
26 membership) may only apply to particular instantiations of the contract template, while other aspects may apply to all
27 instantiations of the contract template. For example, assignment rules and policies can be considered as parameters in
28 a contract template. The style of contract specification determines the method of community establishment, as well as
29 other aspects of the community life-cycle.
- 30

31 7.3.2 Relationships between communities

32 An enterprise specification can describe several communities. A community can be considered in the context
33 of some other community or communities to which it is related. These communities may be related in various
34 ways, including relationships when:

- 35 -- The enterprise object fulfilling a role in one community is part of another community, perhaps
36 fulfilling a role in that community.
- 37 -- A composite enterprise object is part of a community and it or one of its component objects is part
38 of another community.
- 39 -- A c-object of one community is part of another community, fulfilling a certain role in that
40 community.
- 41 -- The specification of two or more communities requires a role in each community to be fulfilled by
42 the same enterprise object; since that object participates in the behaviour of each of those
43 communities, it affects the state of all those communities.

1 -- A portion of the text of the specification of one community is reused in the specification of another
2 community.

3 NOTES:

4 1 – Partitioning may be done because of readability, reuse of specification fragments or interoperability of enterprise
5 objects.

6 2 – When a c-object fulfils a role in another community, the community that object represents (a sub-community)
7 becomes governed by the policy rules of the other community.

8 3 – The population process can be late and dynamic, i.e., a role can be fulfilled by a c-object through a match-making
9 process that considers the sub-community policies, interacting capabilities, interfaces and behaviour description with
10 respect to the requirements stated for the role in the other community.

11 4 - The role and community specifications, as well as type and template specifications, can be private to a specification
12 and development environment, or they can be stored to a repository that can be shared by a wider audience of several
13 development environments and groups.

14 For interaction between two communities to be meaningful, there must be some element of shared objective,
15 which itself implies a higher level of community of which both communities will be members, and a common
16 set of policies will apply.

17 NOTES:

18 1 – The element of shared objective and the common set of policies can be formed either at design time and included
19 in the specifications of the communities or left for run-time negotiation or more straightforward testing of acceptability
20 during community population.

21 2 – The communities involved may have differing policy rules all of which the enterprise object should be able to
22 conform to.

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

25 **7.4 Enterprise object rules**

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

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

31 An enterprise object may be refined as a community at a greater level of detail. All enterprise objects in an
32 enterprise specification fulfil at least one role in at least one community. In fulfilling their roles, enterprise
33 objects, participate in actions, some of which are interactions with other enterprise objects.

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

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

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

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

39 **7.5 Common community types**

40 Three community types are:

41 -- domain

42 -- federation

43 -- ownership community

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

4 NOTE: For example, Internet forms a technical domain that is controlled by the standardization organisation
5 responsible for Internet protocols, while the owners of communication connections and computers in Internet each
6 have an ownership community comprising of a set of equipment."

7 **7.5.1 Domain community type**

8 A community of type <x>-domain contains one "<x>-controller" role and one or more "<x>-controlled" roles,
9 where the controller controls the controlled with regard to the <x>-aspect of their behaviour.

10 NOTE - The core/environment nature of the controlled role is left to specifiers.

11 **7.5.2 Federation community type**

12 A community of type <x>-federation contains two or more <x>-federation member roles which are filled by
13 <x>-domains. The objective of an <x>-federation is to enable the control of the <x>-controlled elements in the
14 individual domains to be shared among the <x>-controllers of those domains. The specific manner in which
15 the <x>-control is shared requires further refinement of the federation community type. <X>-controllers may
16 declare their policies amongst themselves and commit their controlled community to some contract but cannot
17 prescribe policies on each other's controlled communities.

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

20 Editor's Note - Note that "Generally, the controlling object is not a member of the associated
21 domain" or federation. [2-10.3]

22 **7.5.3 Ownership Community Type**

23 The enterprise object filling the controller role (also known as the owner role) must be the owner of the
24 enterprise objects filling each of the controlled roles (also known as the owned roles). The controlled
25 behaviour is all behaviour of the object filling the owned role apart from that which is controlled in other
26 domains or which cannot be subject to control. The owner may either prescribe policies of the community or
27 delegate an authorization to do so to an agent. This authorization may also be withdrawn.

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

30 **7.6 Lifecycle of a community**

31 **7.6.1 Establishing a community**

32 An enterprise specification can include establishing behaviour for a community.

33 The establishing behaviour by which a community is established includes the assignment of enterprise objects
34 to roles. The community contract specifies criteria for deciding which objects are so assigned. The enabled
35 behaviour is consistent with the roles.

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

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

40 **7.6.2 Changes in a community**

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

1 Where an enterprise specification allows changes to the assignment of enterprise objects to roles, a role can
2 become empty. In such a case, the specification must ensure that any obligations associated with the role can
3 continue to be fulfilled.

4 **7.6.3 Terminating a community**

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

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

9 **7.7 Objective rules**

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

11 Every community has exactly one objective. This objective is stated in its contract. An objective can be a
12 composition of sub-objectives.

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

16 A sub-objective may be assigned to a process; in that case, the process is specified to meet the sub-objective
17 and the objective is met by objects performing the actions of the process.

18 Some policies of a community may be criteria for purposeful selection. Such policies result in the selection of
19 behaviour that suits the objective of that community

20 The objective of an enterprise object is consistent with objectives expressed by the role that it fulfils. Hence,
21 where a c-object fulfils a role in a community, the objective of the community of which the c-object is an
22 abstraction is consistent with the objective expressed by this role.

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

25 **7.8 Behaviour rules**

26 **7.8.1 Roles and processes**

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

28 The behaviour of a community is a composition of the expected actions in which the objects of the community
29 participate in fulfilling the roles of the community, together with a set of constraints on when these actions
30 may occur. These constraints may define the possible ordering of these actions and the decomposition of some
31 actions.

32 NOTE – There are many specification styles for expressing the ordering of actions. The modelling language chosen
33 for expressing an enterprise specification may impose certain styles.

34 The assignment of actions to the enterprise objects that comprise a community is defined in terms of roles. A
35 role identifies an abstraction of the community behaviour. All of the actions of that abstraction are associated
36 with the same enterprise object in the community. Each action of the community is either part of a single role
37 behaviour or is an interaction that is part of more than one role behaviour. Each abstraction is labelled as a
38 role. The behaviour identified by that role is subject to the constraints specified in the contract of the
39 community and in the structure of the community. The emphasis is on the enterprise objects that participate in
40 the particular behaviour.

1 Role behaviour decomposes the behaviour of the community into roles that can each be performed by an
2 enterprise object in the community. The enterprise object that performs the role behaviour is said to fulfil that
3 role within the community or is said to be assigned to that role within the community.

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

6 Editor's note – Or involves collective behaviour.

7 The actions and their ordering can be defined in terms of processes. A process identifies an abstraction of the
8 community behaviour that includes only those actions that are related to achieving some particular
9 result/purpose/sub-objective within the community. Each abstraction is labelled with a process name. The
10 emphasis is on what the behaviour achieves.

11 Process behaviour decomposes the behaviour of the community into processes.

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

14 **7.8.2 Role rules**

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

17 An enterprise object may fulfil several roles in one community, and may fulfil roles in several communities. A
18 role in a community may be filled by different objects at different times, but only by one enterprise object at
19 any one time. At any location in time a role in a community may be unfilled, provided that the constraints in
20 the specification of the community so permit.

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

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

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

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

28 NOTE – An enterprise object may come to fulfill a role in a community in other ways than by the instantiation of a
29 template (see subclause 7.1).

30 When an enterprise object is assigned to a role the types of the role should not contradict any of the types
31 satisfied by that object unless the specification includes mechanisms to determine and resolve such
32 inconsistencies.

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

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

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

42 NOTES

43 1 – The concept of role de-couples the expected behaviour from the identities of particular enterprise objects. A role is
44 a placeholder (a formal parameter) providing an identifier for some part of the community behaviour. Associated with

1 a role is a set of constraints on the behaviour expected of any enterprise object that is to fulfil the role; these are
2 requirements on the candidate types of enterprise object.

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

6 2 –If the term ‘<x> object’ is used in an enterprise specification, where <x> is a role, it should be interpreted as
7 meaning ‘an enterprise object fulfilling the role, <x>’. Where an enterprise object fulfils multiple roles, the names
8 can be concatenated.

9
10 Editor's Note - The following two paragraphs were in subclause 7.7.2.2 of the second committee draft.

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

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

16 7.8.3 Interface roles and interactions between communities

17 Objects of a community may interact with objects outside that community to achieve its objectives. This can
18 occur in one of two ways:

- 19 -- either an object outside the community and an object inside the community interact as part of
20 another community;
- 21 -- or the objects outside the community interact with an object (a c-object) that is a composition of
22 some or all of the objects of the community. In this case there are interface roles in the
23 community that identify these interactions.

24 The second case occurs when a community is specified at two different levels of abstraction:

- 25 -- as the configuration of enterprise objects that is the community, and
- 26 -- as a c-object that is an abstraction of the community and is part of a community of wider scope.
27 Then actions in which the c-object participates are identified by interface roles of the
28 community.

29 Temporary Note – In the first case, is there a need to state any constraints that apply between the role of an object in a
30 community and the role of the same object in a second community in which it interacts with an object outside the first
31 community?

32 If the community is represented as a c-object, a mapping should be specified between the interface of the c-
33 object and the interface roles in actual specifications.

34 7.8.4 Enterprise objects and actions

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

- 37 -- The object can participate in carrying out the action; in this case it is said to be an actor with
38 respect to that action.
- 39 -- The object can be mentioned in the action; in this case it is said to be an artefact with respect to
40 that action.
- 41 -- The object can both be essential for the action and require allocation or possibly become
42 unavailable; in this case it is said to be a resource with respect to that action.

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

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

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

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

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

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

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

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

14 NOTES

15 1 – Therefore, roles in a community can be partitioned into actor roles, artefact roles and resource roles with respect to
16 that community.

17 2 – A role is specified in such a way that the behaviour associated with the role, the policies applying to the role, the
18 responsibilities associated with the role, and the relationships with other roles are stated. For example, for each actor
19 role there are complete descriptions of all actions of objects fulfilling that role, and for each action, identification of all
20 the artefact roles mentioned in the action.

21 Editor's Notes

22 1 – 'Complete' is ambiguous in this context.

23 2 – Might the specification of an actor role be abstract, hiding some details.

24 **7.8.5 Process rules**

25 **7.8.5.1 Specifying processes**

26 A process is a directed acyclic graph of steps, in which each step is an action that:

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

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

31 A process is initiated by the occurrence of specified state.

32 A process is terminated by the entry into a specified state.

33 Editor's Note – The bullet points in the first paragraph of this subclause suggests that the
34 second and third paragraphs should end with 'specified state or states.'

35 Note that a state is always the state of an object.

36 NOTES –

37 1 – The use of 'acyclic' indicates that the trace, or history, of steps does not contain cycles of cause and effect. This
38 does not prevent the use of notations with a concept of iteration; such looping concepts generate a sequence of distinct
39 step occurrences.

40 2 – In an enterprise specification, a process is an abstraction of the behaviour of some configuration of objects in which
41 the identities of the some objects have been hidden as a result of the abstraction.

42 Each step need not be performed by the same enterprise object in the community. A step in a process may
43 itself be modelled as a (sub-)process.

1 **7.8.5.2 Mapping between processes and roles**

2 In any mapping between processes and roles each step identified in a process is associated with at least one
3 actor role. Different steps in a process can be associated with different actor roles.

4 **7.9 Policy rules**

5 A policy specification is an identified part of an enterprise specification that is likely to evolve during the
6 lifetime of the ODP system or that can be changed to tailor a single specification to apply to a range of
7 different ODP systems.

8 A policy specification includes:

- 9 -- the name of the policy;
- 10 -- the rules;
- 11 -- the elements of the specification affected by the policy;
- 12 -- constraints on changing the rules;
- 13 -- rules for changing the policy.

14 Editor's Note – Are the last two items the same? The policy is the set of rules. “Policy: A set
15 of rules related to a particular purpose.” [2-11.2.7] (Some) rules are constraints. “Not every
16 policy is a constraint.” [2-11.2.7]

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

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

21 **7.9.1 General policy rules**

22 A community specification can include policies that apply to the community as a whole.

23 The policies specified in a community specification are directly related to the achievement of the objective of
24 the community.

25 As a result of the creation of a community the policies that apply to the enterprise objects that are its members
26 can be changed in the following ways:

- 27 -- the enterprise objects, by being subject to the agreement which creates the community, are subject
28 to its policies and their behaviour is subject to different constraints from those that applied
29 before;
- 30 -- the enterprise objects may be subject to obligations not to do things unless specifically allowed by
31 the rules of the community;

32 **7.9.2 Obligations, permissions and prohibitions**

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

36 **7.9.2.1 Obligation**

37 An obligation is defined by:

- 38 -- a behaviour;
- 39 -- a role or roles involved in that behaviour;

1 -- a predicate on behaviour;

2 -- an authority that imposes the obligation.

3 If a set of enterprise objects is subject to the obligation then, when the predicate is true, there is an obligation
4 on the set of objects to participate in the behaviour, by order of the authority.

5 A standing obligation is an obligation for which the predicate on the behaviour is always true

6 **7.9.2.2 Permission**

7 A permission is defined by:

8 -- a behaviour

9 -- a role or roles involved in that behaviour

10 -- a predicate on behaviour

11 -- an authority which grants the permission

12 If a set of enterprise objects has this permission, then, when the predicate is true, that set of enterprise objects
13 is allowed to take part in the action when fulfilling the roles, by order of the authority.

14 Editor's Note – Does 'allowed' convey the meaning of permission? The following may be
15 more precise:

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

19 **7.9.2.3 Prohibition**

20 Like a permission, a prohibition is defined by:

21 -- a behaviour

22 -- a role or roles involved in that behaviour

23 --a predicate on behaviour

24 -- an authority which imposes the prohibition

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

27 Editor's Note – Might 'cannot' be too strong here? The following text may be more precise:

28 If a set of enterprise objects has this prohibition, then, when the predicate is true, there is an
29 obligation for that set of enterprise objects not to take part in the action when fulfilling the
30 roles, by order of the authority.

31

32 Editor's Note - Is a subclause wanted here, prescribing the form to be used to specify an
33 authorization?

34 **7.9.3 Nesting of policy frameworks**

35 The enterprise specification must indicate the possible nesting of communities and thus all the policy rules
36 governing the behaviour of the enterprise objects fulfilling roles in a community. An enterprise object must
37 conform to all policies in all communities it participates.

38 The enterprise specification must also indicate whether possible dynamic changes in policy rules of the
39 community (for instance, caused by federation negotiations) should be enforced on subcommunities or peer

1 communities. A community may require that the peer communities co-operating with it follow the same
2 method of adopting dynamic changes to policies as it itself follows.

3 Note: In practice, this currently means static policies in each community, but a c-object may inherit its policy
4 requirement from several outer communities.

5 **7.9.4 Policy violations**

6 TEMPORARY NOTE – The working group considers that text in this subclause should be rewritten using the concept,
7 violation and invites National Body contributions of appropriate text.

8 Policies can be specified as policed and enforced, or unpoliced.

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

10 Pessimistic enforcement is preventative and requires the specification of mechanisms to ensure that the right
11 things are done and the wrong things are not done.

12 Pessimistic enforcement is specified when trust is low (i.e. when non-compliance is expected to be rife) and
13 the damage caused by non-compliance is potentially high, and when viable preventative mechanisms can be
14 created and/or effective sanctions can be applied post-hoc when non-compliance occurs.

15 Optimistic enforcement is not preventative. It requires the specification of mechanisms to detect and
16 report/correct non-compliance.

17 Optimistic enforcement is specified when trust is high and the potential damage due to non-compliance is low,
18 and when viable preventative mechanisms do not exist.

19 **7.9.5 Organisation of policy**

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

25 The specification of policy covers

- 26 -- limits on the community behaviour arising from the permission, or otherwise, of the enterprise
27 objects to take part in interactions; this can be expressed by statements that the various
28 enterprise objects have the capability to perform particular sets of actions;
- 29 -- the degree to which, and the circumstances in which, one enterprise object can take over the role
30 and responsibilities of another, acting as a substitute for it.

31 An enterprise specification may require that successful performance of an interaction between enterprise
32 objects of a community requires that a set of permissions exists. If an enterprise specification requires
33 permissions they are either:

- 34 -- associated with a particular target role in that interaction, or
- 35 -- associated with that interaction as a whole.

36 Editor's Note – The meaning of 'target role' may not be clear.

37 When an enterprise specification requires permissions for an interaction, if permission for that interaction is
38 missing, the interaction fails and therefore the enterprise object may fail to fulfill its role.

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

41 Editor's Note – Does this mean subject to the permission rules of this subclause?

1 **7.10 Force rules**

2 The environment of an ODP system includes parties. Parties can have intentions and are responsible for their
3 actions.

4 The concepts of subclause 6.4 are used to model an action of a party or parties to which an intention is
5 ascribed. In particular, an act specifies behaviour suited, according to some criteria, to an objective.

6 Acts of parties are not predetermined by the specification of an ODP system. However, the specification will
7 identify the acts of parties that a machine is prepared to participate in, respond to or record. The specification
8 will also identify the acts that a machine is prepared to participate in as an agent of a party. An enterprise
9 specification describes the authority delegated to an ODP system in terms of:

- 10 -- the parties that have delegated authority to the system;
- 11 -- the authority that each party has delegated;
- 12 -- the duration and conditions of the delegation;
- 13 -- provisions for additional delegation and withdrawal of delegation during the operation of the
14 system.

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

18 An enterprise specification may also specify delegation from any party to another party or machine (including
19 a whole ODP system or one of its parts). Likewise, an enterprise specification may specify delegation, insofar
20 as provided in delegation by a party, by a machine to a party or other machine.

21 Each action of an ODP system pursuant to a delegation is an action selected to suit the objectives of a
22 principal, and is therefore an act.

23 An enterprise specification specifies the force of each act.

24 **7.10.1 Delegation rules**

25 A principal is responsible for the act of an object acting as its agent.

26 When a machine is an agent with authority to delegate, and it may (subject to the provisions of that authority)
27 delegate (authority, responsibility, a function, ...) to one of its parts, to a machine not one of its parts or to a
28 party. That object is then an agent of the principal and has the same authority as if delegated directly by the
29 principal.

30 **7.10.2 Authority rules**

31 For each authority delegated, an enterprise specification describes the force of acts of an agent in exercising
32 that authority. The authority delegated may be:

- 33 -- to make a commitment; this binds the principal
- 34 -- to issue a declaration; this establishes the truth of some proposition just as if the principal had
35 made the declaration;
- 36 -- to further delegate an authority to a party or machine (or, if the agent is a machine, to part of the
37 agent); this causes the agent delegated to have the authority;
- 38 -- to make a prescription that establishes a rule; such a rule has the same force as if the principal had
39 made the prescription.

1 **7.10.3 Commitment rules**

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

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

7 **7.10.4 Declaration rules**

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

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

13 **7.10.5 Prescription rules**

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

- 15 -- that object is a party that by its nature may establish rules,
- 16 -- that object is an agent of such a party, delegated authority to establish rules on behalf of that party,
- 17 -- the specification explicitly provides for those actions of that object that will be prescriptions, or
- 18 -- that object is, in a previous epoch, specified to establish rules.

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

21 **8 Consistency rules**

22 **8.1 Viewpoint correspondences**

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

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

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

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

4 There are two kinds of standardization requirements relating to correspondences:

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

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

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

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

22 **8.2 Enterprise and information specification correspondences**

23 **8.2.1 Concepts related by correspondences**

24 The enterprise concepts related are:

- 25 -- Objective;
- 26 -- Community;
- 27 -- Enterprise object;
- 28 -- Scope;
- 29 -- Role;
- 30 -- Actor;
- 31 -- Artefact;
- 32 -- Process;
- 33 -- Policy.

34 The information concepts related are:

- 35 -- Information object;
- 36 -- Dynamic schema;
- 37 -- Static schema;
- 38 -- Invariant schema.

39 **8.2.2 Required correspondences**

40 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
41 additional correspondence rules in this sub-clause.

42 There are no required correspondences.

1 **8.2.3 Required correspondence statements**

2 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
3 additional correspondence rules in this sub-clause.

4 The specifier shall provide:

- 5 -- for each enterprise object in the enterprise specification, a list of those information objects that
6 describe part or all of its state;

7 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for n:m
8 correspondences.

- 9 -- for each role in each community in the enterprise specification, a list of those information object
10 types that describe part or all of the state associated with the roles it fills;

- 11 -- for each policy in the enterprise specification, a list of the invariant, static and dynamic schemata
12 of information objects modified by the policy; an information object is included if it is
13 associated with the enterprise community that is subject to that policy;

- 14 -- for each action in the enterprise specification, the information objects that have a dynamic schema
15 constraining that action;

16 TEMPORARY NOTE - The working group invites National Body comment and contributions on whether this
17 constraint is necessarily a definition of the action.

- 18 -- for each relationship between enterprise objects, the invariant schema that represents it.

19 **8.3 Enterprise and computational specification correspondences**

20 **8.3.1 Concepts related by correspondences**

21 The enterprise concepts related are:

- 22 -- Enterprise object;
23 -- Actor role;
24 -- Enterprise interaction;
25 -- Policy.

26 The computational concepts related are:

- 27 -- Computational object behaviour;
28 -- Computational interaction;
29 -- Computational interface;
30 -- Binding object.

31 **8.3.2 Required correspondences**

32 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
33 additional correspondence rules in this sub-clause.

34 There are no required correspondences.

35 **8.3.3 Required correspondence statements**

36 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
37 additional correspondence rules in this sub-clause.

38 The specifier shall provide:

- 39 -- for each interaction in the enterprise specification, a list of those computational interfaces and
40 operations or streams that correspond to the enterprise action, together with a statement of

- 1 whether this correspondence applies to all occurrences of the action, or is qualified by a
2 predicate;
- 3 -- for each role affected by a policy in the enterprise specification, a list of the computational object
4 types that exhibit choices in the computational behaviour that are modified by the policy;
- 5 -- for each interaction between roles in the enterprise specification, a list of computational binding
6 types that are constrained by the enterprise interaction;
- 7 -- for each enterprise interaction type, a list of computational behaviour types capable of
8 representing (i.e. acting as a carrier for) the enterprise interaction type.

9 **8.4 Enterprise and engineering specification correspondences**

10 **8.4.1 Concepts related by correspondences**

11 The enterprise concepts related are:

- 12 -- Enterprise object;
- 13 -- Principal role;
- 14 -- Actor role;
- 15 -- Policy

16 The engineering concepts related are:

- 17 -- Basic engineering object;
- 18 -- Node;
- 19 -- Stub;
- 20 -- Binder;
- 21 -- Protocol object;
- 22 -- Interceptor.

23 **8.4.2 Required correspondences**

24 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
25 additional correspondence rules in this sub-clause.

26 There are no required correspondences.

27 **8.4.3 Required correspondence statements**

28 TEMPORARY NOTE - The working group invites National Body comment and contributions on the need for
29 additional correspondence rules in this sub-clause.

30 The specifier shall provide:

- 31 -- for each enterprise object in the enterprise specification, a list of those engineering nodes that
32 support some or all of its behaviour;
- 33 -- for each interaction between roles in the enterprise specification, a list of engineering channel
34 types and stub, binding or protocol objects that are constrained by the enterprise interaction;

35 **9 Relations between standards and product development**

36 TEMPORARY NOTE - The working group invites National Body to propose another title for this clause.

1 **9.1 Compliance, conformance, testing and consistency**

2 This standard follows the normal practice in standardization work in using the term compliance to describe
3 the relationship between two standards. One standard complies with another if it makes correct use of the
4 ideas, vocabulary or framework defined there. This implies that, if a specification is compliant, directly or
5 indirectly, with some other specifications, then the propositions which are true in those specifications are also
6 true in a conformant implementation of the specification.

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

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

17 ODP systems are specified in terms of a number of viewpoints, and this gives rise to an accompanying
18 requirement for consistency between the different viewpoint specifications. The key to consistency is the idea
19 of correspondences between specifications; i.e., a statement that some terms or structures in one specification
20 correspond to other terms and structures in a second specification. Correspondences can be identified between
21 two different viewpoint specifications in a single language or in two different languages. Statements of
22 correspondences between two languages imply equivalent correspondences between any pair of specifications
23 expressed in those languages.

24 **9.2 Completeness**

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

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

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

41 **9.3 Scoping statement**

42 Any specification should include a scoping statement that specifies the preconditions for the use of that
43 specification. A scoping statement says whether a specification is appropriate in a given situation, and must be

1 satisfied before it makes sense to make observations of the real world and compare these with specified
2 observable properties to test conformance to the specification.

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

6 The scoping statement expresses the agreed scope of the system being specified, in terms of what it is intended
7 to do, and what properties the environment must have for the system to operate.

8 The scope of the system is dependent on choices made by the specifier. Each specification needs to be
9 considered separately for completeness, compliance and conformance, as the specifier finds appropriate under
10 the prevailing circumstances.

11 **10 Enterprise Language Compliance**

12 An enterprise specification compliant with this Recommendation | International Standard shall use the
13 concepts defined in clause 6 and those in clause 5.1 of ITU-T Rec. X.903 | ISO/IEC 10746-3, as well as the
14 concepts defined in ITU-T Rec. X.902 | ISO/IEC 10746-2, structured as specified in clause 7.

15 Concepts from ITU-T Rec. X.902 | ISO/IEC 10746-3 not refined in this Recommendation | International
16 Standard may also be employed. Where such concepts are employed, the specification concerned shall include
17 explanations of the relationships between the concepts concerned and those defined in clause 6.

18 Concepts from other modelling languages may also be employed. Where such concepts are employed, the
19 specification concerned shall include or refer to definitions of each such concept, in terms of the concepts
20 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
21 10746-3 , and explanations of the relationships between such concepts and those defined in clause 6.

22 **11 Conformance and reference points**

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

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

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

37 The Enterprise Language places requirements on organizational structures and business processes that cannot
38 be observed directly, but must be deduced from the variety of interactions between the system or systems
39 involved and their environment. In claiming conformance to an enterprise specification, the system provider
40 must state what observable reference points in the system are conformance points, and how observations at
41 these points can be interpreted to correspond to enterprise concepts. With this information, a tester of the
42 system is in a position to determine by observation whether the system behaves correctly. In ODP,

1 conformance is based on the declaration of Engineering Viewpoint reference points (in clauses 5-7 of ITU-T
2 Rec. X.903 | ISO/IEC 10746-3), and the implementor of an Enterprise Specification must state
3 correspondences to the engineering viewpoint in order to relate observations at the engineering reference
4 points to enterprise concepts.
5

1

2 **Annex A Overall structure of an enterprise specification**

3 This is a part of a meta-model of the enterprise viewpoint. This meta-model defines the structure of an
4 enterprise specification.

5 NOTES:

6 1 – This meta-model is not complete, but it is helpful for the understanding of the enterprise viewpoint.

7 2 – Some links that come from RM-ODP Part 2 are not defined in this meta-model (i.e. the fact that a role is an
8 identifier for a behaviour, et cetera).

9 3 – This meta-model is compliant with the ODP Type Repository Function (ISO/IEC FDIS 14769: Information
10 Technology – Open Distributed Processing – Type Repository Function).

11 4 – The notation used in this meta-model is UML (ISO/IEC DIS 19501-1, Information Technology – Unified Modeling
12 Language (UML) - Part 1: Specification).

13

1 Index

- 2
- 3 <X> domain, **4**
- 4 <X> federation, **5**
- 5 <X> template, **4**
- 6 act, **9**
- 7 actioj, **9**
- 8 action, **4, 7, 8, 9**
- 9 activity, **7**
- 10 activity structure, **7**
- 11 actor, **8**
- 12 actor role, **8**
- 13 agent, **8, 9, 22**
- 14 artefact, **8**
- 15 artefact role, **8**
- 16 authorisation, **8, 9**
- 17 authority, **8**
- 18 behaviour, **7**
- 19 behaviour, **8, 9, 18**
- 20 c-object, **7**
- 21 commitment, **9, 22**
- 22 community, **5, 8, 18**
- 23 composite object, **4**
- 24 composition, **4**
- 25 computational interface, **5**
- 26 configuration (of objects), **4**
- 27 conformance, **4**
- 28 conformance point, **4**
- 29 contract, **4, 9**
- 30 contracting party, **9**
- 31 declaration, **9, 22**
- 32 delegate, **8**
- 33 delegation, **8, 9**
- 34 dynamic schema, **5**
- 35 entity, **4, 7**
- 36 environment (of an object), **4**
- 37 environment contract, **4**
- 38 epoch, **4**
- 39 establishing behaviour, **22**
- 40 incremental modification, **4**
- 41 instance (of a type), **4**
- 42 instantiation (of an <X> template), **4**
- 43 interface, **4**
- 44 interface role, **8**
- 45 internal action, **4**
- 46 interworking reference point, **4**
- 47 invariant, **4**
- 48 invariant schema, **5**
- 49 liaison, **4, 22**
- 50 location in space, **4**
- 51 location in time, **4**
- 52 machine, **22**
- 53 object, **4**
- 54 objective (of an <X>), **7**
- 55 obligation, **4, 9, 22**
- 56 ODP standards, **4**
- 57 ODP system, **4**
- 58 owner (of an <X>), **8**
- 59 party, **7, 8, 9, 22**
- 60 perceptual reference point, **4**
- 61 permission, **4, 9**
- 62 policy, **4, 5, 9**
- 63 postcondition, **4**
- 64 precondition, **4**
- 65 prescription, **9, 22**
- 66 principal, **8, 22**
- 67 process, **7**
- 68 programmatic reference point, **4**
- 69 prohibition, **4, 9**
- 70 proposition, **4**
- 71 purpose (of a system), **7**
- 72 purposeful selection, **9**

- 1 quality of service, **4**
- 2 reference point, **4**
- 3 refinement, **4**
- 4 resource, **8**
- 5 resource role, **8**
- 6 responding object, **4**
- 7 role, **4, 8**
- 8 rule, **9, 22**
- 9 s-community, **7**
- 10 scope, **7**
- 11 state (of an object), **4**
- 12 static schema, **5**
- 13 step, **7**
- 14 subtype, **4**
- 15 system, **4, 7**
- 16 template, **7**
- 17 type, **7**
- 18 type (of an <X>), **4**
- 19 valuation, **9**
- 20 viewpoint (on a system), **4**
- 21 violation, **9**