INTERNATIONAL STANDARD

ISO/IEC 15414

Third edition
2015-04-15

Information technology — Open distributed processing — Reference model — Enterprise language

Technologies de l'information — Traitement réparti ouvert — Modèle de référence — Langage d'entreprise

Reference number
ISO/IEC 15414:2015(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 15414 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in collaboration with ITU-T. The identical text is published as ITU-T X.911 (09/2014).

This third edition cancels and replaces the second edition (ISO/IEC 15414:2006), which has been technically revised.

Introduction

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

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

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

0.1 RM-ODP

The RM-ODP consists of:

- Part 1: Rec. ITU-T X.901 | ISO/IEC 10746-1: **Overview**: This contains a motivational overview of ODP, giving scoping, justification and explanation of key concepts, and an outline of the ODP architecture. It contains explanatory material on how the RM-ODP is to be interpreted and applied by its users, who may include standards writers and architects of ODP systems. It also contains a categorization of required areas of standardization expressed in terms of the reference points for conformance identified in ITU-T Rec. X.903 | ISO/IEC 10746-3. This part is informative.
- Part 2: Rec. ITU-T X.902 | ISO/IEC 10746-2: **Foundations**: This contains the definition of the concepts and analytical framework for normalized description of (arbitrary) distributed processing systems. It introduces the principles of conformance to ODP standards and the way in which they are applied. This is only to a level of detail sufficient to support Rec. ITU-T X.903 | ISO/IEC 10746-3 and to establish requirements for new specification techniques. This part is normative.
- Part 3: Rec. ITU-T X.903 | ISO/IEC 10746-3: **Architecture**: This contains the specification of the required characteristics that qualify distributed processing as open. These are the constraints to which ODP standards shall conform. It uses the descriptive techniques from Rec. ITU-T X.902 | ISO/IEC 10746-2. This part is normative.
- Part 4: Rec. ITU-T X.904 | ISO/IEC 10746-4: **Architectural semantics**: This contains a formalization of the ODP modelling concepts defined in Rec. ITU-T X.902 | ISO/IEC 10746-2 clauses 8 and 9. The formalization is achieved by interpreting each concept in terms of the constructs of one or more of the different standardized formal description techniques. This part is normative.
- Rec. ITU-T X.911 | ISO/IEC 15414: **Enterprise language**: this Recommendation | International Standard.

0.2 Overview and motivation

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

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

The purpose of this Recommendation | International Standard is to:

- Refine and extend the enterprise language defined in Rec. ITU-T X.903 |ISO/IEC 10746-3 to enable full enterprise viewpoint specification of an ODP system.
- Explain the correspondences of an enterprise viewpoint specification of an ODP system to other viewpoint specifications of that system.
- Ensure that the enterprise language, when used together with the other viewpoint languages, is suitable for the specification of a concrete application architecture to fill a specific business need.

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

This Recommendation | International Standard provides a common language (set of terms and structuring rules) to be used in the preparation of an enterprise specification capturing the purpose, scope and policies for an ODP system. An enterprise specification is a part of the specification of an ODP system using viewpoints defined by Recommendation ITU-T X.903 | ISO/IEC 10746-3. The specification of the ODP system can describe any or all of:

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

The primary audience for this Recommendation | International Standard is those who prepare and use such specifications. The audience includes ODP system owners and users, including subject management experts, and developers and maintainers of ODP systems, tools and methodologies.

The motivation for the enterprise language is to support standardized techniques for specification. This improves communication and helps create consistent specifications.

The preparation of specifications often falls into the category referred to as analysis or requirement specification. There are many approaches used for understanding, agreeing and specifying systems in the context of the organizations of which they form a part. The approaches can provide useful insights into both the organization under consideration and the requirements for systems to support it, but they generally lack the rigour, consistency and completeness needed for thorough specification. The audiences of the specifications also vary. For agreement between the potential users of an ODP system and the provider of that system, it may be necessary to have different presentations of the same system – one in terms understood by clients, and one in terms directly related to system realization.

The use of enterprise specifications can be wider than the early phases of the software engineering process. A current trend is to integrate existing systems into global networks, where the functionality of interest spans multiple organizations. The enterprise language provides a means to specify the joint agreement of common behaviour of the ODP systems within and between these organizations. The enterprise specification can also be used in other phases of the system life cycle. The specification can, for example, be used at system run-time to control agreements between the system and its users, and to establish new agreements according to the same contract structure. Enterprise viewpoint specifications may contain rules for inter-organizational behaviour.

This Recommendation | International Standard also provides a framework for the development of software engineering methodologies and tools exploiting ODP viewpoint languages, and a set of concepts for the development of enterprise viewpoint specification languages. For these purposes, this Recommendation | International Standard provides rules for the information content of specifications and the grouping of that information. Further requirements on the relationships between enterprise language concepts and concepts in other viewpoints are specific to the methodologies, tools or specification languages to be developed.

An enterprise specification defines the purpose, scope, and policies of an ODP system and it provides a statement of conformance for system implementations. The purpose of the system is defined by the specified behaviour of the system while policies capture further restriction on the behaviour between the system and its environment or within the system itself related to the business decisions by the system owners.

An enterprise specification also allows the specification of an ODP system that spans multiple domains and is not owned by a single party, and specification of the collective behaviour of a system that is divided into independently specified and independently working subsystems.

This generality places greater emphasis on the expression of correct or normal behaviour and on the chains of responsibility involved in achieving it. For example, the advent of service oriented and cloud computing has led to the need to specify business rules and behaviour in a way that clearly describes obligations, permissions, authorizations and prohibitions, as well as the accountability of each of the objects involved in an enterprise specification. This involves the expression of the so-called deontic aspects of the behaviour of the system, and of the accountability of the objects involved.

Annex A presents a metamodel of the enterprise language, illustrating the key concepts of the enterprise language and their relationships. This annex is normative. Annex B provides examples using the concepts and structuring rules of the enterprise language and provides examples of how they may be used. Annex C indicates how the semantics of deontic constraints may be expressed. Annexes B and C are informative.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Open distributed processing – Reference model – Enterprise language

1 Scope

This Recommendation | International Standard provides:

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

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

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

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

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

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

2 Normative references

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

2.1 Identical ITU-T Recommendations | International Standards

– Recommendation ITU-T X.902 (2009) | ISO/IEC 10746-2:2010, *Information technology – Open Distributed Processing – Reference Model: Foundations*.

– Recommendation ITU-T X.903 (2009) | ISO/IEC 10746-3:2010, *Information technology – Open Distributed Processing – Reference Model: Architecture*.

– Recommendation ITU-T X.904 (1997) | ISO/IEC 10746-4:1998, *Information technology – Open Distributed Processing – Reference Model: Architectural semantics*.

– Recommendation ITU-T X.906 (1997) | ISO/IEC 19793:2012, *Information technology – Open distributed processing – Use of UML for ODP system specifications.*

2.2 Additional References

– ISO/IEC 19505-2:2012, *Information Technology – Object Management Group Unified Modelling Language (OMG UML) – Part 2: Superstructure*.

3 Terms and definitions

3.1 Definitions from ODP standards

3.1.1 Modelling concept definitions

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

- action;
- activity;
- behaviour (of an object);
- composite object;
- composition;
- configuration (of objects);
- conformance;
- conformance point;
- contract;
- <X> domain;
- entity;
- environment contract;
- environment (of an object);
- epoch;
- establishing behaviour;
- event;
- instantiation (of an <X> template);
- internal action;
- invariant;
- liaison;
- location in time;
- name;
- object;
- obligation;
- ODP standards;
- ODP system;
- permission;
- policy;
- policy declaration;
- policy envelope;
- policy setting behaviour;
- policy value;
- prohibition;
- proposition;
- reference point;
- refinement;
- role;
- service;
- state (of an object);
- subsystem;

- subtype;
- system;
- <X> template;
- terminating behaviour;
- type (of an <X>);
- viewpoint (on a system).

3.1.2 Viewpoint language definitions

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

- binder;
- capsule;
- channel;
- cluster;
- community;
- computational behaviour;
- computational binding object;
- computational object;
- computational interface;
- computational viewpoint;
- dynamic schema;
- engineering viewpoint;
- enterprise object;
- enterprise viewpoint;
- <X> federation;
- information object;
- information viewpoint;
- interceptor;
- invariant schema;
- node;
- nucleus;
- operation;
- protocol object;
- static schema;
- stream;
- stub;
- technology viewpoint;
- <viewpoint> language.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

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

5 Conventions

This Recommendation | International Standard contains references to Parts 2 and 3 of the RM-ODP and to the normative text of this Recommendation | International Standard. Each reference is of one of these forms:

- [Part 2-n.n] – a reference to clause n.n of RM-ODP Part 2: Foundations, X.902 | ISO/IEC 10746-2;
- [Part 3-n.n] – a reference to clause n.n of RM-ODP Part 3: Architecture, X.903 | ISO/IEC 10746-3;
- [n.n] – a reference to clause n.n of this Recommendation | International Standard.

For example, [Part 2-9.4] is a reference to Part 2 of the reference model, (Rec. ITU-T X.902 | ISO/IEC 10746-2), clause 9.4 and [6.5] is a reference to clause 6.5 of this Recommendation | International Standard. These references are for the convenience of the reader.

This Recommendation | International Standard also contains some text which is a modification of text from Part 3 of the reference model, Rec. ITU-T X.903 | ISO/IEC 10746-3. Such text is marked by a reference like this: [see also 3-5.n]. The modifications are authoritative with respect to the enterprise language.

6 Concepts

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

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

The grouping into subclauses and the headings of the subclauses of this clause are informative.

6.1 System concepts

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

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

6.2 Community concepts

6.2.1 objective (of an <X>): Practical advantage or intended effect, expressed as preferences about future states.

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

NOTE 2 – In the text of Rec. ITU-T X.903 | ISO/IEC 10746-3 [Part 3-5] the terms *purpose* and *objective* are synonymous. The enterprise language emphasizes the term *objective* and emphasizes the need to express an objective in measurable terms.

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

6.3 Behaviour concepts

6.3.1 active enterprise object: An enterprise object that is able to fill an action role. In other words, it is an enterprise object that can be involved in some behaviour.

NOTE – The behaviour of active enterprise objects is constrained by deontic and accountability concepts, defined in clauses 6.4 and 6.6. The deontic tokens defined in clause 6.4 are not themselves active enterprise objects.

6.3.2 actor (with respect to an action): A role (with respect to that action) in which the enterprise object fulfilling the role participates in the action. That object may be called an actor.

NOTE – It may be of interest to specify which actor initiates that action.

6.3.3 artefact (with respect to an action): A role (with respect to that action) in which the enterprise object fulfilling the role is referenced in the action. That object may be called an artefact.

NOTE 1 – An enterprise object that is an artefact in one action can be an actor in another action.

NOTE 2 – The object filling an artefact role in an action is an active enterprise object being referenced in the action and this should not be confused with the way a deontic token held by an object involved in the action constrains its performance.

6.3.4 resource (with respect to an action): A role (with respect to that action) in which the enterprise object fulfilling the role is essential to the action, requires allocation, or may become unavailable. That object may be called a resource.

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

NOTE 2 – A consumable resource object may become unavailable after some amount of use. Any resource object may become unavailable after some amount of time (for example, if a duration or expiry time has been specified for the resource).

6.3.5 **interface role**: A role in a community, identifying behaviour which takes place with the participation of objects that are not members of that community.

6.3.6 **process**: A collection of steps taking place in a prescribed manner.

NOTE 1 – The prescribed manner may be a partially ordered sequence of steps.

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

NOTE 3 – A process may have multiple end points.

NOTE 4 – An enterprise specification may define types of process and may define process templates.

NOTE 5 – A process is an abstraction of a behaviour, and so shares any objectives defined for that behaviour.

NOTE 6 – A process specification can be a workflow specification.

6.3.7 **step**: An abstraction of an action, used in a process, that may leave unspecified some or all of the objects that participate in that action.

6.3.8 **violation**: A behaviour contrary to that required by a rule.

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

6.4 Deontic concepts

6.4.1 **deontic token**: An enterprise object which expresses a constraint on the ability of an active enterprise object holding it to perform certain actions. An active enterprise object carries a set of deontic tokens, which control the occurrence of conditional actions within its behaviour. These tokens are either permits, burdens or embargos. A deontic token is not itself an active enterprise object; it is held by exactly one active enterprise object.

NOTE – The constraint is expressed by a rule forming part of the token; an appropriate notation for expressing this rule will be selected by the specifier. The notation allows the declaration of the active enterprise object and conditional action to which it applies, and requirements on other enterprise objects fulfilling roles in the conditional action controlled. For example, the rule may control the performance of a purchase action by a consumer, and place restrictions on the supplier and the artefact being purchased. The notation may also declare periods of validity or deadlines for performance of the action. The kind of associated information allowed will depend on whether the token is a permit, a burden or an embargo.

6.4.2 **token group**: A group of tokens named so that it can be referred to as a whole.

NOTE – A notation for expressing deontic rules will provide the means for declaring and naming groups of deontic tokens. Changes that result, for example, from the performance of speech acts can then be applied to complete groups of tokens without the need to reference all the group members individually.

6.4.3 **burden**: A deontic token encapsulating the statement of an obligation on the active enterprise object holding it, thereby modifying the urgency of the active enterprise object in performing associated conditional actions within its behaviour.

6.4.4 **embargo**: A deontic token encapsulating the statement of a prohibition on the active enterprise object holding it, thereby modifying the ability of the active enterprise object to perform associated conditional actions within its behaviour.

6.4.5 **permit**: A deontic token encapsulating the statement of a permission on the active enterprise object holding it, thereby modifying the ability of the active enterprise object to perform associated conditional actions within its behaviour.

6.4.6 **conditional action**: An action which has associated preconditions based on the sets of burdens, permits and embargos carried by the active enterprise objects filling its various action roles. The specification of the conditional action states what permits are required for, what burdens favour, and what embargos inhibit performance of the action.

6.4.7 **speech act**: An action whose performance results in a change to the sets of deontic tokens (permits, embargos and burdens) carried by the active enterprise objects filling its various action roles. A speech act may result in the addition of new tokens to the performer of an action role, or in the removal of tokens from the performer of an action role, or the transfer of tokens from the performer of one action role to the performer of another action role in the same interaction.

NOTE 1 – Many actions for which parties are accountable are speech acts; examples are prescription and commitment.

NOTE 2 – Although we speak informally of a speech act as changing or transferring a token, it is more precise to describe this process as the destruction of one of the token existing before the act occurred and the construction of a new token based on the information available when the act is performed. The definition of the speech act type includes the formal rules governing the content and location of the token that is generated. Transfer of a token by a speech act is therefore the process of destruction of a token held by one of the participating objects followed by construction of a new token with the same contents at its destination.

6.5 Policy concepts

6.5.1 **policy**: A constraint on a system specification foreseen at design time, but whose detail is determined subsequent to the original design, and capable of being modified from time to time in order to manage the system in changing circumstances. A policy is expressed as a rule, which may, in turn, be a composition of several sub-rules. A policy is introduced into a specification by a policy declaration. At any point in time it has a particular policy value, but the policy value can be changed by a defined policy setting behaviour, so long as it remains within a defined policy envelope. [See 2-11.2.8 to 2-11.2.12]

NOTE – A rule can be expressed as an obligation, an authorization, a permission or a prohibition. Not all the constraints involved restrict the behaviour; for example, some policies may represent an empowerment.

6.5.2 **affected behaviour**: A fragment of behaviour (including an action, step or process) that is constrained by the current policy value.

6.6 Accountability concepts

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

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

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

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

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

NOTE 1 – The enterprise objects participating in an action of commitment may be parties or agents acting on behalf of a party or parties. In the case of an action of commitment by an agent, the principal responsible for the agent becomes obligated.

NOTE 2 – The fact that an enterprise object is obligated is expressed by associating with it a burden describing the obligation.

6.6.3 **prescription**: An action that establishes a rule.

6.6.4 **authorization**: An action indicating that a particular behaviour shall not be prevented.

NOTE 1 – Unlike a permission, an authorization is an empowerment.

NOTE 2 – The fact that an enterprise object has performed an authorization is expressed by it issuing a required permit and itself undertaking a burden describing its obligation to facilitate the behaviour.

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

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

6.6.6 **delegation**: The action that assigns something, such as authorization, responsibility or provision of a service to another object.

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

6.6.7 **evaluation**: An action that assesses the value of something.

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

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

6.6.8 **agent**: An active enterprise object that has been delegated something (authorization, responsibility, provision of a service, etc.) by, and acts for, a party (in exercising the authorization, carrying out the responsibility, providing the service, etc.).

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

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

NOTE 3 – A specification may state that, in its initial state, an active enterprise object is an agent of a party.

6.6.9 **principal**: A party that has delegated something (authorization, provision of a service, etc.) to another.

7 Structuring rules

This clause refines and extends the structuring rules defined in clause 5.2 of Rec. ITU-T X.903 | ISO/IEC 10746-3, as they apply to the concepts of community, enterprise object, objective, behaviour and policy. It defines structuring rules for the accountability concepts defined in clause 6.6. It uses the concepts defined in ITU-T Rec. X.902 | ISO/IEC 10746-2, in clause 5.1 of Rec. ITU-T X.903 | ISO/IEC 10746-3 and in clause 6.

7.1 Overall structure of an enterprise specification

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

NOTE 1 – The environment of an ODP system and the ODP system itself may span multiple organizations. More than one party may own the ODP system.

NOTE 2 – An enterprise specification may specify the collective behaviour of separately specified and interworking subsystems of the ODP system.

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

The enterprise specification includes, within the areas of interest of the specification users, the objective and scope of the ODP system, the policies for the ODP system (including those of any environment contracts), the community in which the ODP system is specified and the roles fulfilled by the ODP system and other enterprise objects in that community, and the processes in which the ODP system and enterprise objects in its environment participate.

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

NOTE 3 – This minimal enterprise specification describes the objective and scope of the ODP system; this description is necessary for completeness of the enterprise specification.

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

NOTE 4 – The set of communities in an enterprise specification may include, for example, communities at both more abstract and more detailed levels than the minimal enterprise specification, as well as communities relating to functional decomposition of the ODP system and to ownership of the ODP system and its parts.

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

NOTE 5 – This may represent, for example, a federation.

The scope of the system is defined in terms of its intended behaviour; in the enterprise language this is expressed in terms of roles, processes or policies, and the relationships of them. Reference to behaviour in general includes both basic behaviour, in terms of processes, steps and actions, and any associated deontic or accountability mechanisms.

NOTE 6 – It may be meaningful to discuss the intended, delivered or expected scope of a system in various phases of planning, development or deployment. In such cases, the term "scope" should be appropriately qualified.

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

7.2 Contents of an enterprise specification

An enterprise specification is structured in terms of the elements explained in clause 7.1 and the other concepts identified in clause 6, as well as the relationships between them.

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

- the characteristics of the element; or
- the type or types of the element; or
- a template for the element.

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

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

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

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

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

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

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

7.3 Community rules

7.3.1 Community

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

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

A community contract constrains the existence or behaviour of its enterprise object members in such a way as to satisfy the objectives of the community. The community objective may be decomposed into sub-objectives and the behaviour decomposed into distinct sub-behaviours that each satisfies one of the sub-objectives.

In general, the enterprise specification defines the behaviour which sets up the community contracts and each contract specifies the behaviour necessary to terminate it. However, a specification may cover only a certain period of time throughout which the community and its contract exist. In such cases, their existence forms part of the initial state of the specification, creation being implicit.

An enterprise specification may reuse an existing contract template, incorporating it as a specification fragment by reference. In some cases, the contract may define new enterprise objects that are to be created as part of the community creation, or the community may be formed from pre-existing enterprise objects. The creation of a new community will, in general, place obligations, permissions and prohibitions on its founder members.

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

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

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

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

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

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

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

NOTE 2 – Types of communities may be related by refinement.

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

NOTE 4 – The specification of a community may include specific enterprise objects, relationships between those objects, and relationships of those object to enterprise objects assigned to roles in that community.

NOTE 5 – The concept of a contract is particularly important within a community specification, because the contract contains all the information about the structure of a community, its behaviour, and the way it operates.

7.3.2 Relationships between communities

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

When composing *communities*, there will be a set of *policies* common to those *communities*. These *policies* shall be consistent, although unspecified *behaviour* in the composite *community* may allow room for (mutually inconsistent) *behaviour* in each individual *community*.

Communities may interact in the following ways:

- a community object fulfils one or more roles in other communities;
- two or more community objects interact in fulfilling roles in some other community;
- the enterprise specification requires the same object to fulfil specific roles in more than one community and the behaviour of the object in any given role may affect its behaviour in other roles;
- an object, in fulfilling an interface role (see clause 7.8.3) of one community, interacts with an object fulfilling an interface role in another community;
- a community includes behaviour for creating new communities.

NOTE 1 – For example, federation establishment means creation of a new community, which involves putting in place the contract of the community, including the structure and policies for that community.

NOTE 2 – For more information about interactions involving community objects and the communities they represent, see clause 7.8.3 – Interface roles and interactions between communities.

For each of these ways of interacting there is an invariant that determines the constraints on the collective behaviour of the communities concerned. In all kinds of interactions of communities it is critical to consider the invariants that determine the constraints on the collective behaviour of the communities concerned, and the objectives and policies that govern the different communities. The communities involved in an interaction may have differing rules; all of the objects participating in that interaction must be able to conform to all those rules.

These invariants include:

- where a community object fulfils one or more roles in another community, the community that the community object represents is governed by the policies of the other community;
- where two or more community objects interact in fulfilling roles in some other community, the communities that the community objects represent are related by those interactions;
- where the same object is required to fill specific roles in more than one community, an invariant specifies how the actions of that object affect those communities;

- where the same object is required to fill specific roles in more than one community, that object becomes governed by the policies of all those communities;
- where two or more communities interact, there is a set of policies common to those communities.

NOTE 3 – Where two communities interact, an implicit community may be considered, such that the community objects representing both communities are members of and are governed by the policies of that community. The element of shared objective and the common set of policies can be formed either at design time and included in the specifications of the communities or left for run-time negotiation or testing of acceptability during community population.

NOTE 4 – The communities involved may have differing rules; an enterprise object must be able to conform to all these rules.

7.4 Enterprise object rules

An enterprise specification will include enterprise objects; an enterprise object is any object in an enterprise specification. Active enterprise objects and the entities they model are those felt to be necessary or desirable to specify the system from the enterprise viewpoint or to understand the enterprise specification. Deontic tokens are enterprise objects that express constraints on the behaviour of the active enterprise objects holding them.

NOTE 1 – An active enterprise object may be a model of a human being, a legal entity, an information processing system, a resource or a collection or part of any of these.

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

In fulfilling their roles in communities, enterprise objects participate in actions, some of which are interactions with other enterprise objects. The behaviour of an enterprise object is restricted by the roles to which it is assigned.

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

- by design the community includes the object;
- the object becomes a member of the community at the time of creation of that community; or
- the object becomes a member of the community as a result of dynamic changes in the configuration of the community.

NOTE 2 – The contract of the community includes rules for the assignment of enterprise objects to roles; thus, to establish a community it is not necessary to identify the enterprise objects of that community.

NOTE 3 – The contract of the community can include rules that change the community structure (for example, the number of roles).

7.5 Common community types

Two common community types are:

- <X>-domain.
- <X>-federation.

Communities of these types can be specified so that they overlap totally or partially. These basic community types do not imply any hierarchical relationships. A specification may choose to use these community types, but it does not have to do so.

7.5.1 <X>-domain community type

An <X>-domain community comprises an <X>-domain of enterprise objects in the roles of controlled objects and an enterprise object in the role of controlling object for the <X>-domain. The <X>-domain community establishes the characterizing relationship <X> between the enterprise objects in the roles of controlled objects and the enterprise object in the role of controlling object [Part 2-10.3].

7.5.2 <X>-federation community type

An <X>-federation community is a community of a number of pre-existing communities cooperating to achieve a shared objective. Each member of a federation agrees by participating in the federation to be bound by the contract of the community (which may include obligations to contribute resources or to constrain behaviour) so as to pursue the shared objective. At the same time, a federation preserves the autonomy of the original participants. The specification of a federation may hide any aspects of the members not directly relevant to the shared objective; it may include defined behaviour that enables a participant to withdraw from the federation at any time.

7.6 Life cycle of a community

7.6.1 Establishing a community

An enterprise specification can include establishing behaviour for a community.

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

7.6.2 Assignment policy

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

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

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

Members of the community may be selected on demand according to the assignment policy for that community.

The rules of the assignment policy can directly identify the objects, or may use a supporting mechanism with more complex assignment rules. The rules may be based on object identifiers, relationships between objects, object capabilities, technologies, preceding commitments, object behaviour, etc. Objects may need to possess appropriate deontic tokens to fulfil the intended role.

7.6.3 Changes in a community

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

The changes to be considered here include:

- addition, change, and removal of policies or rules;
- addition, change, and removal of roles;
- addition and removal of enterprise objects;
- addition, change, and removal of processes or steps.

NOTE – Changes to a community shall maintain the overall consistency of the contract of that community.

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

If an enterprise object ceases to fulfil the role to which it is assigned according to an assignment rule, that object violates the contract of the community.

7.6.4 Terminating a community

An enterprise specification can include terminating behaviour for a community.

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

NOTE 2 – Some communities are permanent and never terminate.

7.7 Objective rules

Every community has exactly one objective. The objective is expressed in a contract which specifies how the objective can be met.

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

The purpose of an ODP system is expressed as one or more objectives (or sub-objectives) of the community or set of communities in which the ODP system fulfils roles. If the ODP system is itself modelled as a community, then the purpose of the system is the objective of that community.

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

The policies of a community restrict the community behaviour in such a way that it is possible to meet the objective. At any particular time, the policy value in force is always within the policy envelope, which is chosen so that the objective is always achievable; see clause 07.9. Such policies result in behaviour that suits the objective of the community.

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

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

7.8 Behaviour rules

7.8.1 Roles and processes

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

NOTE 1 – There are many specification styles for expressing when actions may occur (e.g., sequencing, preconditions, partial ordering, etc.). The modelling language chosen for expressing an enterprise specification may impose certain styles.

NOTE 2 – Part 2 of the Reference Model provides the concept of an activity [Part 2-8.6]. It also provides concepts for specifying the structure of an activity [Part 2-13.1], including chains, threads, forking actions, joining actions and spawning actions. These can be used to structure community behaviour.

Community roles are used to decompose the behaviour of the community into parts that can each be performed by a particular enterprise object in the community. The role is a formal placeholder for associating actions in the community behaviour with the enterprise object which is to perform them. The enterprise object that performs the behaviour of a role is said to fulfil that role within the community or is said to be assigned to that role within the community. All of the actions of that role are associated with the same enterprise object in the community. Each action of the community is either part of a single role behaviour or is an interaction that is part of more than one role behaviour. Each of these abstractions is labelled as a role. The behaviour identified by that role is subject to the constraints specified in the contract and structure of the community. In contrast to the specification of actions and their ordering in terms of processes (see below), the emphasis is on the enterprise objects that participate in the particular behaviour.

Each action will be part of at least one role, but can be part of many roles (when the action involves an interaction and the various action roles involved in it are performed by different community roles).

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

Processes decompose the behaviour of the community into steps.

NOTE 3 – The choice of using a role-based or process-based modelling approach will depend on the modelling method used and the aim of modelling. A combination of the two approaches may be used.

7.8.2 Role rules

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

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

NOTE 1 – If the term '<X> object' is used in an enterprise specification, where <X> is a role, it should be interpreted as meaning 'an enterprise object fulfilling the role, <X>'. Where an enterprise object fulfils multiple roles, the names can be concatenated.

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

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

NOTE 2 – Examples are the modelling of the members of a committee, or the modelling of the customers of a service.

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

NOTE 3 – Enterprise specifications may refer to existing mechanisms for determining and resolving incompatibilities between types of objects and requirements set by roles, thus enlarging the set of objects acceptable for a given role.

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

NOTE 4 – The constraints of the community must be satisfied throughout its lifetime. However, these invariants may change, which may determine different epochs in this lifetime. Such changes may lead to changes in the sets of roles and in the sets of relationships between roles of the community.

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

NOTE 5 – The rules define what the object to fulfil a role shall be capable of doing, without being restricted by earlier commitments, and what relationships to other objects are required or prohibited.

7.8.3 Interface roles and interactions between communities

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

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

- as a configuration of enterprise objects, where some of these objects fulfil interface roles; and
- as a community object that is an abstraction of the community. Interactions in which that community object can participate as part of some other community are identified by the interface roles of the community which that community object represents.

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

7.8.4 Enterprise objects and actions

A way of categorizing the involvement of an enterprise object in an action is to consider it as having an action role with respect to that action:

- The object can participate in carrying out the action; in this case, it is said to fulfil an actor role, or to be an actor with respect to that action.
- The object can be mentioned in the action; in this case, it is said to fulfil an artefact role, or to be an artefact with respect to that action.
- The object can both be essential for the action and require allocation or possibly become unavailable during or after the action; in this case, it is said to fulfil a resource role, or to be a resource with respect to that action.

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

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

NOTE 3 – In this clause, the concept of role is used in the context of the community in which a role is specified. Thus an object's role is an identifier for some behaviour that the object exhibits in that community. In certain circumstances, the behaviour identified is a specific action; in such cases, this is explicitly specified.

An actor in an action can also be an artefact with respect to that action. Likewise, an actor in an action can also be a resource with respect to that action (if it itself is used in the action). When a resource is essential for an action, the action is constrained by the availability of that resource.

Like any other actions, these actions are also constrained by the possession of deontic tokens by the enterprise objects fulfilling its action roles. If the action is a speech act, the action will result in the modification of the set of deontic tokens held by each object participating in it.

7.8.5 Process rules

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

A process is a collection of steps taking place in a prescribed manner. A step may be associated with multiple roles. Every step shall have one or more actors.

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

The collective behaviour of a community may be represented as a set of processes. This set can be seen as a more abstract process performed by a single role fulfilled by a community object. Also, a step of a process can be further refined as a more detailed process. This refinement may also involve the refinement, or introduction, of deontic tokens.

NOTE – A process, or the behaviour from which it is abstracted, will often be traceable to an objective or sub-objective of the community that contains it. However, some fine grain processes may represent housekeeping tasks defined by the specifier rather than derived directly from the declared community objective.

7.8.6 Behaviour violations

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

NOTE 1 – These may arise, for example, in a federation where there is not full control of the interacting objects, or in other situations where an action is not considered to be essential enough to be specified in detail for all possible participants of an interaction.

NOTE 2 – An enterprise specification may include a rule prescribing types of actions to be taken by an object in the event of certain types of violations. That rule is an obligation, which applies to that object. Failure to take the prescribed actions is a violation of that rule.

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

7.8.7 Deontic token rules

The specification of enterprise behaviour typically involves the expression of deontic constraints such as obligations, permissions and prohibitions. These are incorporated into an object-based model by introducing enterprise objects called deontic tokens. If an active enterprise object has an associated deontic token, then the corresponding deontic constraint applies to the object's behaviour. However, deontic tokens are not themselves active enterprise objects and are not directly involved in interactions by taking action roles. Each deontic token is associated with exactly one active enterprise object. There are three types of deontic token:

- a burden represents an obligation on the objects with which it is associated;
- a permit represents a permission held by the objects with which it is associated;
- an embargo represents a prohibition affecting the objects with which it is associated.

These deontic constraints are created or modified by specific types of action which are called speech acts. A speech act may result in the creation of deontic tokens or the transfer of such tokens between objects playing particular action roles in the speech act. The destruction of a token at the end of its life cycle is also generally performed by a speech act, although tokens may destroy themselves as a result of a timeout or other trigger. The rules for transfer, creation or destruction of tokens are expressed by the action type of the speech act (see Note 2 in clause 6.4.7).

NOTE 1 – The set of tokens held by the objects concerned determines whether a speech act can take place and what its consequences are. For example:

- it may be necessary for an object to hold a permit before it can perform a speech act;
- having an embargo may prevent an object from performing a speech act, even though the action would otherwise be permitted by the object's role;
- a burden held by an object may be discharged as a result of its performing a speech act;
- the performance of a delegation speech act may transfer a group of tokens (for example, burdens and permits) to the object to which responsibility is delegated.

NOTE 2 – Deontic constraints may be used to express business rules. For example, the action of ordering goods may be controlled by the purchaser needing to have a suitable permit, and may result in the purchaser holding a burden representing the obligation to pay for them within a stated period. The life cycle of the payment burden starts with the goods order speech act and ends with the payment speech act. Between these two actions, the burden may also be transferred between a number of active enterprise objects, such as agents or subcontractors.

NOTE 3 – When an object fills a community role, some tokens associated with the community as a whole will be transferred to the object by virtue of its taking up the role. Similarly, when an object leaves a role, some tokens revert to the community as a whole, which must have procedures for handling them. The actions of filling or leaving a role are therefore speech acts. For example, a lecturer may have a burden requiring her to give a lecture course, but, if she resigns, the department has the obligation to have the course given. If a new lecturer is appointed, she receives the burden to give the course when she fills her new role.

NOTE 4 – These rules can be modelled in a number of ways. For one approach to formal specification, see Annex C.

A deontic token may be in either an active or a pending state. When it is in an active state, the constraint it carries is applied to control the behaviour of the active enterprise object that holds it. However, when it is in the pending state, this constraint is masked so that it does not affect the current behaviour.

NOTE 5 – A speech act may place a deontic token in a pending state in situations where the effect of that token need to be suspended but may need to be restored. For example, a party may attempt to discharge an obligation by placing an equivalent obligation on one of its agents; in this situation, it may retain the original obligation in a pending state. If the agent discharges the obligation, the party's burden is also discharged, but if the agent fails, the exception raised may be specified to return the original burden from the pending to the active state. Similarly, a delegated permit may be retained in a pending state by its original holder so that it cannot be used while the delegation is in effect but reverts to active form if the delegate fails.

NOTE 6 – If the active object with which a deontic token is associated is a community object, it may be assigned to a specific community role within the community object. When this role is filled by an enterprise object, the token can be cloned and the copy associated with the role-filling object. The original token is placed in a pending state. If the role ceases to be filled, the original token again becomes active. The token associated with the role-filling object is deleted if the association of the original token with the community object is broken for any reason.

7.8.8 The specification of obligations, permissions, prohibitions and authorizations

The following subclauses provide a way of specifying deontic rules:

7.8.8.1 Obligation

An obligation is modelled as a burden deontic token and is defined by:

- a set of rules that prescribes the obligation;
- an identified behaviour that is subject to that set of rules;
- a role or roles involved in that behaviour that are subject to the set of rules;
- a subset of that behaviour that is required to occur;
- optionally, an object or objects that may fulfil the roles involved.

When the obligation applies, the enterprise objects fulfilling the roles that are subject to the set of rules shall engage in the required behaviour.

A standing obligation is an obligation that always applies.

NOTE – The fact that a certain action results in an obligation is derived from a set of rules that applies to the situation. This may be a piece of legislation, a social norm, a contract or some other convention.

7.8.8.2 Permission

A permission is modelled as a permit deontic token and is defined by:

- an authorization domain that prescribes the permission;
- an identified behaviour that is subject to that domain;
- a role or roles involved in that behaviour that are subject to the domain;
- a subset of that behaviour that is allowed to occur;
- optionally, an object or objects that may fulfil the roles involved.

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

NOTE 1 – There is, however, no guarantee that the action succeeds. For example, the action may have participants in other domains in which the action is prohibited.

An enterprise specification may specify that interactions between enterprise objects of a community may occur only when permission for the interaction exists. Such permissions may apply either to a particular role in the interaction or to the interaction as a whole.

NOTE 2 – In such cases, if a permission required for an interaction is missing, that interaction fails and, therefore, the enterprise objects may fail to fulfil their roles.

7.8.8.3 Prohibition

A prohibition is modelled as an embargo deontic token and is defined by:

- an authorization domain that prescribes the prohibition;
- an identified behaviour that is subject to that domain;
- a role or roles involved in that behaviour that are subject to the domain;
- a subset of that behaviour that shall not occur;
- optionally, an object or objects that may fulfil the roles involved.

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

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

7.8.8.4 Authorization

An authorization is modelled using a combination of permit and burden deontic tokens. In it, an active enterprise object, acting as an authority, grants a permit to be held by an authorized agent and, as a result, that authority has an obligation to empower the authorized agent by removing other restrictions that might prevent use of the permit. The permit held by the authorized agent is defined by:

- an authorization domain that prescribes the authorization;
- an identified behaviour that is subject to that domain;
- a role or roles involved in that behaviour that are subject to the domain;
- a subset of that behaviour that is allowed to occur;
- optionally, an object or objects that may fulfil the roles involved.

The burden on the authority is defined by:

- a set of rules that prescribes the obligation;
- an identified behaviour that is subject to that set of rules;
- a role or roles involved in that behaviour that are subject to the set of rules;
- a subset of that behaviour that is required to occur;
- optionally, an object or objects that may fulfil the roles involved.

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

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

7.9 Policy rules

7.9.1 The specification of a policy

A policy identifies the specification of a behaviour, or constraints on a behaviour, that can be changed during the lifetime of the ODP system or that can be changed to tailor a single specification to apply to a range of different ODP systems. Changes in the policies of a community during its lifetime can occur only if an enterprise specification includes behaviour that can cause such changes. The specification of a policy includes the declaration of a policy envelope that bounds the changes in policy that are to be allowed; at any point in time, there is a single policy value associated with the policy, and this value must satisfy the constraints given by the policy envelope.

Policies may apply to a community as a whole, to enterprise objects that fulfil roles in that community (regardless of which role), to roles (i.e., to all actions named by those roles), or to action types. They may also apply to the collective behaviour of a set of enterprise objects.

The specification of a policy includes:

- the name of the policy;
- the rules it applies, expressed as obligations, permissions, prohibitions and authorizations;
- the elements of the enterprise specification affected by the policy;
- the policy envelope that constrains the possible restrictions or behaviours that are acceptable as policy values;
- behaviour for changing the policy;
- a default policy value to be used until any explicit initial change takes place.

The specification of policy may cover the degree to which, and the circumstances in which, there can be delegation by one enterprise object to another.

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

NOTE 2 – For a given *policy envelope*, only one *policy value* is in force at a particular point in time. This policy value may be selected from a set of values defined in the policy envelope or it may be a statement in a policy language that is consistent with constraints in the policy envelope.

NOTE 3 – Policy may, for example, be used to configure generic objects to apply them in a specific situation, or to express a pervasive decision that affects many objects.

NOTE 4 – Policies may cover, for example, rules about:

– behaviour (such as processes);

– qualities of service;

– the names or types of objects with which a given object may interact;

– the technology by means of which interactions may be performed.

NOTE 5 – Policies for a community may be composed from other policies; other communities may be subject to the same policies. Policies may be specified in a community template; the template may include parameters used in establishing policies.

NOTE 6 – Policies for a community may form part of a hierarchy of policies. This may position the community within a larger environment, for example, with respect to some organizations.

NOTE 7 – Policies for a community are established when the community is specified or when it is established according to the specified establishing behaviour. The establishing behaviour may involve other, already established, communities or the controlling objects of <X> domain communities.

An enterprise object shall conform to all policies in each community in which it participates.

When an enterprise object of a community fulfils a role in another community, the policies of the two communities that apply to that object might conflict. Where an enterprise object is subject to policies of more than one community, the enterprise specification shall ensure that policy conflicts do not exist, or specify how policy conflicts are to be prevented or discovered and resolved, or state that policy conflicts are allowed to cause failures.

NOTE 8 – Examples of how an enterprise specification may specify how conflict is to be prevented or resolved include specification of a policy governing the assignment of roles to objects where such conflict may arise, specification of a policy prescribing modification of behaviour, and specification of a mechanism for modification of conflicting policies.

7.9.2 Policies for federation

Establishing an <X> federation community involves establishing a set of policies for that community. An enterprise object in the <X> federation community shall conform both to the policies of the <X> domain community to which it belongs and to the policies of the <X> federation community.

NOTE 1 – In an inter-organizational environment, the policies for each domain community and for the federation community may have separate life cycles.

NOTE 2 – The mechanisms for conflict management may be supported by the specification language or the runtime environment of the systems involved, and may thus not necessarily be visible explicitly in the enterprise specification.

NOTE 3 – Examples of policy conflict cases are:

a) (Specification-time assurance) The specifications of the federation communities are compared and any conflicts are resolved in the specifications.

b) (Run-time prevention) Forming federations with policy conflicts is prevented by checking consistency of policies while assigning objects to roles in the federation community.

c) (Run-time discovery and resolution) The federation community includes a behaviour to resolve the conflict by changing policies.

d) (Failure handling) The specification of the federation community provides sanctions or alternative behaviour for cases where behaviour has failed because of policy conflicts.

7.9.3 Policy setting behaviour

The behaviour for changing the policy may include behaviour that changes the rules expressing the policy value or behaviour that replaces the policy value with a different named policy.

NOTE 1 – The behaviour may include constraints on changing that policy.

NOTE 2 – There may be no behaviour for changing the policy (that is, the policy is not changed during the lifetime of the community).

NOTE 3 – Behaviour to negotiate and change policies may be necessary to enable formation of an <X> federation.

7.9.4 Policy enforcement

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

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

Pessimistic enforcement is preventative and requires the specification of mechanisms to ensure that the obligated actions occur, prohibited actions do not occur, and authorized actions are not prevented. Pessimistic enforcement is specified when trust is low (i.e., when non-compliance is expected) and the damage caused by non-compliance is potentially high,

and when viable preventative mechanisms can be created or effective sanctions can be applied after non-compliance occurs.

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

7.10 Accountability rules

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

Parties can have intentions and are accountable for their actions. The concepts of clause 6.6 are used to model an action that involves accountability of a party.

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

7.10.1 Delegation rules

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

- the parties that have delegated authorization to the system;
- the authorization that each party has delegated;
- the duration and conditions of the delegation;
- provisions for additional delegation and withdrawal of delegation during the operation of the system.

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

The delegation may be modelled in terms of a speech act that transfers relevant deontic tokens to the agent and creates additional burdens expressing associated reporting and monitoring responsibilities to be assigned to the principal.

Where delegation by a party to an agent has taken place, an enterprise specification may specify further delegation by that agent to another active enterprise object, so that it also becomes an agent.

7.10.2 Authorization rules

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

- to make a commitment; this binds the principal;
- to issue a declaration; this establishes the truth of a proposition just as if the principal had made the declaration;
- to make a prescription that establishes a rule; such a rule has the same force as if the principal had made the prescription;
- to further delegate an authorization; this causes the agent receiving the further delegation to have the authorization.

7.10.3 Commitment rules

An enterprise specification identifies, for every commitment, the obligation created. It identifies, for every commitment made by an agent, the principals obligated. The commitment can be modelled as creating an appropriate burden deontic token associated with the agent.

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

7.10.4 Declaration rules

A declaration identifies the changes that take place in the environment of an object as the result of an internal action of that object. An enterprise specification defines the conditions required for a particular declaration to be effective. In general, this will require the object to have an appropriate permit deontic token.

NOTE – A declaration may not be effective (cause the intended change in the environment of the object) until there has been an interaction of the object, such as a publication.

7.10.5 Prescription rules

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

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

An important special case of delegation is where the authorized action is a prescription, so that the delegation enables an enterprise object to make a prescription. The delegation then transfers to the object a permit deontic token that enables the action of prescription. Performing the prescription may also create an associated burden to oversee the consequences of the prescription.

8 Compliance, completeness and field of application

8.1 Compliance

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

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

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

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

8.2 Completeness

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

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

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

8.3 Field of application

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

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

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

9 Enterprise language compliance

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

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

10 Conformance and reference points

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

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

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

In claiming conformance to an enterprise specification, the system provider shall state what observable reference points in the system are conformance points, and how observations at these points can be interpreted to correspond to enterprise concepts. With this information, a tester of the system is in a position to determine, by observation, whether the system behaves correctly. In ODP, conformance is based on the declaration of engineering viewpoint reference points (in clauses 5-7 of Rec. ITU-T X.903 | ISO/IEC 10746-3), and the implementer of an enterprise specification shall state correspondences to the engineering viewpoint in order to relate observations at the engineering reference points to enterprise concepts.

11 Consistency rules

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

11.1 Viewpoint correspondences

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

The specifications produced from different ODP viewpoints are each complete statements in their respective viewpoint languages, using their own locally significant names, and so cannot be related without additional information in the form of correspondence statements. What is needed is a set of statements that make clear how constraints from different viewpoints apply to particular elements of a single system to determine its overall behaviour. The correspondence

statements are statements that relate the viewpoint specifications, but do not form part of any one viewpoint specification. The correspondences can be established in two ways:

- by declaring correspondences between terms in two different viewpoint languages, stating how their meanings relate. This implies that the two languages are expressed in such a way that they have a common, or at least a related, set of foundation concepts and structuring rules. Such correspondences between languages necessarily entail correspondences relating to all things of interest which the languages are used to model (e.g., things modelled by objects or actions);
- by considering the extension of terms in each language, and asserting that particular entities being modelled in the two specifications are in fact the same entity. This relates the specifications by identifying which observations need to be interpretable in both specifications.

There are two kinds of standardization requirements relating to correspondences:

- Some correspondences are required in all ODP specifications; these are called required correspondences. If the correspondence is not valid in all instances in which the concepts related occur, the specification is not a valid ODP specification.
- In other cases, there is a requirement that the specifier provides a list of items in two specifications that correspond, but the content of this list is the result of a design choice; these are called required correspondence statements.

The minimum requirement for consistency in a set of specifications for an ODP system is that they exhibit the correspondences defined in the Reference Model [Part 3-10], those defined here, and those defined within the specification itself.

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

11.2 Enterprise and information specification correspondences

11.2.1 Concepts related by correspondences

The enterprise concepts related are:

- community;
- enterprise object;
- enterprise action;
- role;
- policy;
- deontic token.

The information concepts related are:

- information object;
- dynamic schema;
- static schema;
- invariant schema.

11.2.2 Required correspondences

There are no required correspondences.

11.2.3 Required correspondence statements

The specifier shall provide:

- for each enterprise object in the enterprise specification, a list of those information objects (if any) that represent information or information processing concerning the entity represented by that enterprise object;
- for each role in each community in the enterprise specification, a list of those information object types (if any) that specify information or information processing of an enterprise object fulfilling that role;
- for each policy in the enterprise specification, a list of the invariant, static and dynamic schemata of information objects (if any) that correspond to the enterprise objects to which that policy applies; an information object is included if it corresponds to the enterprise community that is subject to that policy;

- for each action in the enterprise specification, the information objects (if any) subject to a dynamic schema constraining that action;
- for each relationship between enterprise objects, the invariant schema (if any) which constrains objects in that relationship;
- for each relationship between enterprise roles, the invariant schema (if any) which constrains objects fulfilling roles in that relationship;
- for each deontic token in the enterprise specification, the information objects (if any) which represent the associated obligation, permission or prohibition and the information objects (if any) which support any associated accountability and traceability.

11.3 Enterprise and computational specification correspondences

11.3.1 Concepts related by correspondences

The enterprise concepts related are:

- enterprise object;
- role;
- enterprise interaction;
- policy;
- deontic token.

The computational concepts related are:

- computational object;
- computational behaviour;
- computational binding object;
- computational interface;
- operation;
- stream.

11.3.2 Required correspondences

There are no required correspondences.

11.3.3 Required correspondence statements

The specifier shall provide:

- for each enterprise object in the enterprise specification, that configuration of computational objects (if any) that realizes the required behaviour;
- for each interaction in the enterprise specification, a list of those computational interfaces and operations or streams (if any) that correspond to the enterprise interaction, together with a statement of whether this correspondence applies to all occurrences of the interaction, or is qualified by a predicate;
- for each role affected by a policy in the enterprise specification, a list of the computational object types (if any) that exhibit choices in the computational behaviour that are modified by the policy;
- for each interaction between roles in the enterprise specification, a list of computational binding object types (if any) that are constrained by the enterprise interaction;
- for each enterprise interaction type, a list of computational behaviour types (if any) capable of carrying out an interaction of that enterprise interaction type;
- for each deontic token in the enterprise specification, the computational rules (if any) necessary to police the correct interpretation of the token.

11.4 Enterprise and engineering specification correspondences

11.4.1 Concepts related by correspondences

The enterprise concepts related are:

- behaviour;
- enterprise object;

- interaction;
- policy;
- role.

The engineering concepts related are:

- binder;
- capsule;
- channel;
- cluster;
- interceptor;
- node;
- nucleus;
- protocol object;
- stub.

11.4.2 Required correspondences

There are no required correspondences.

11.4.3 Required correspondence statements

The specifier shall provide:

- for each enterprise object in the enterprise specification, the set of those engineering nodes (if any) with their nuclei, capsules and clusters, all of which support some or all of its behaviour;
- for each interaction between roles in the enterprise specification, a list of engineering channel types and stubs, binders, protocol objects and interceptors (if any) that are constrained by the enterprise interaction.

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

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

11.5 Enterprise and technology specification correspondence

In accordance with clause 15.5 of Rec. ITU-T X.902 | ISO/IEC 10746-2 and clause 5.3 of Rec. ITU-T X.903 | ISO/IEC 10746-3, an implementer provides, as part of the claim of conformance, the chain of interpretations that permits observation at conformance points to be interpreted in terms of enterprise concepts. While there may be specific correspondences between enterprise policies and technology viewpoint specifications that require the use of particular technologies, there are neither required correspondences nor required correspondence statements.

NOTE – Although there are no required viewpoint correspondences between enterprise viewpoint and technology viewpoint specifications, there may be cases where part of enterprise viewpoint specification has a direct relationship with a technology viewpoint specification or a choice of technology. Such examples include enterprise policy covering performance (e.g., response time), reliability and security.

Annex A

Model of the enterprise language concepts

(This annex forms an integral part of this Recommendation | International Standard.)

This annex gives a normative abstract model of the main concepts from the enterprise language and the relationships between those concepts. The selection of elements in this model, which is expressed in the diagrams below, is chosen to give a basic overview of the main features of the language, even though this results in some redundancy of expression from a formal modelling point of view. In the interests of simplicity, the rich web of relationships with all the supporting concepts from RM-ODP Part 2 has also been left out of the diagrams. In particular, relations with the concepts of type and template are not specifically illustrated.

NOTE 1 – The notation used in this model is UML (ISO/IEC 19505-2:2012 Information Technology – OMG Unified Modelling Language – Superstructure).

NOTE 2 – The convention for expressing an association is such that it can be read as: "Each (or an instance of) <Class X> <verb phrase> <cardinality> <Class Y>", where the verb phrase expresses the role played by an <X> in its relationship with a <Y>, and is placed as a RoleEndName at the <X> end of the association (in accordance with the rules for UML notation in ISO/IEC 19505-2). For example, in Figure A.1 the association that an ODP System has with Scope can be read as "Each ODP system has (the) expected behaviour defined in exactly one Scope". The choice of "each" or "an instance of" will depend on the cardinalities of the association.

NOTE 3 – Where UML role names are not shown explicitly, they are derived from the associated class name, replacing the initial capital with lower case. Where cardinalities are not shown explicitly, they have the value 1.

The diagrams representing this model are presented below under several broad headings, covering the concepts found in clause 6 of this Recommendation | International Standard, namely:

- System concepts – representing the relationships between an enterprise specification and the system that it describes.
- Community concepts – representing relationships between the main enterprise language concepts used in modelling the roles and communities.
- Behavioural concepts – representing additional concepts used in expressing behaviour.
- Policy concepts – supporting the dynamic refinement and modification of an enterprise specification.
- Deontic and accountability concepts – expressing the obligations, permissions and prohibitions associated with enterprise behaviour and supporting traceability from objectives to solutions.
- The life cycle of the deontic token, relating changes to it with the other concepts.

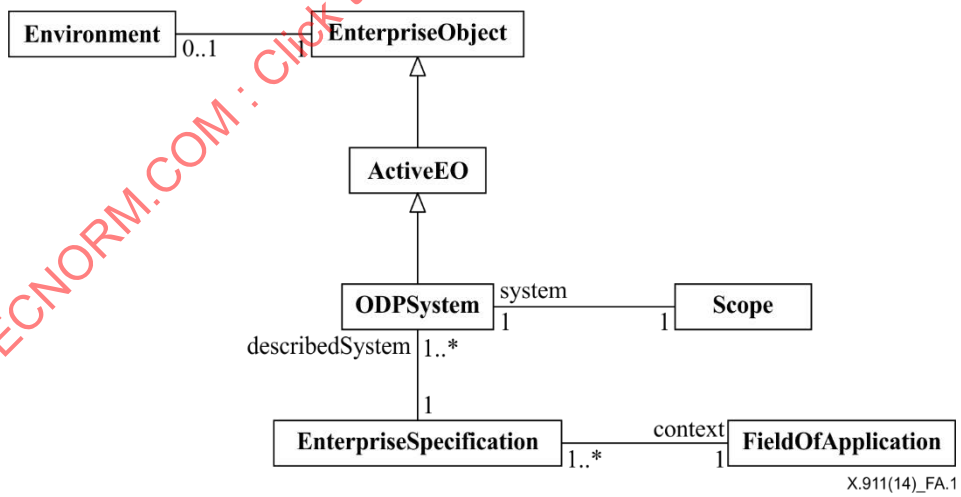

Figure A.1 – System concepts

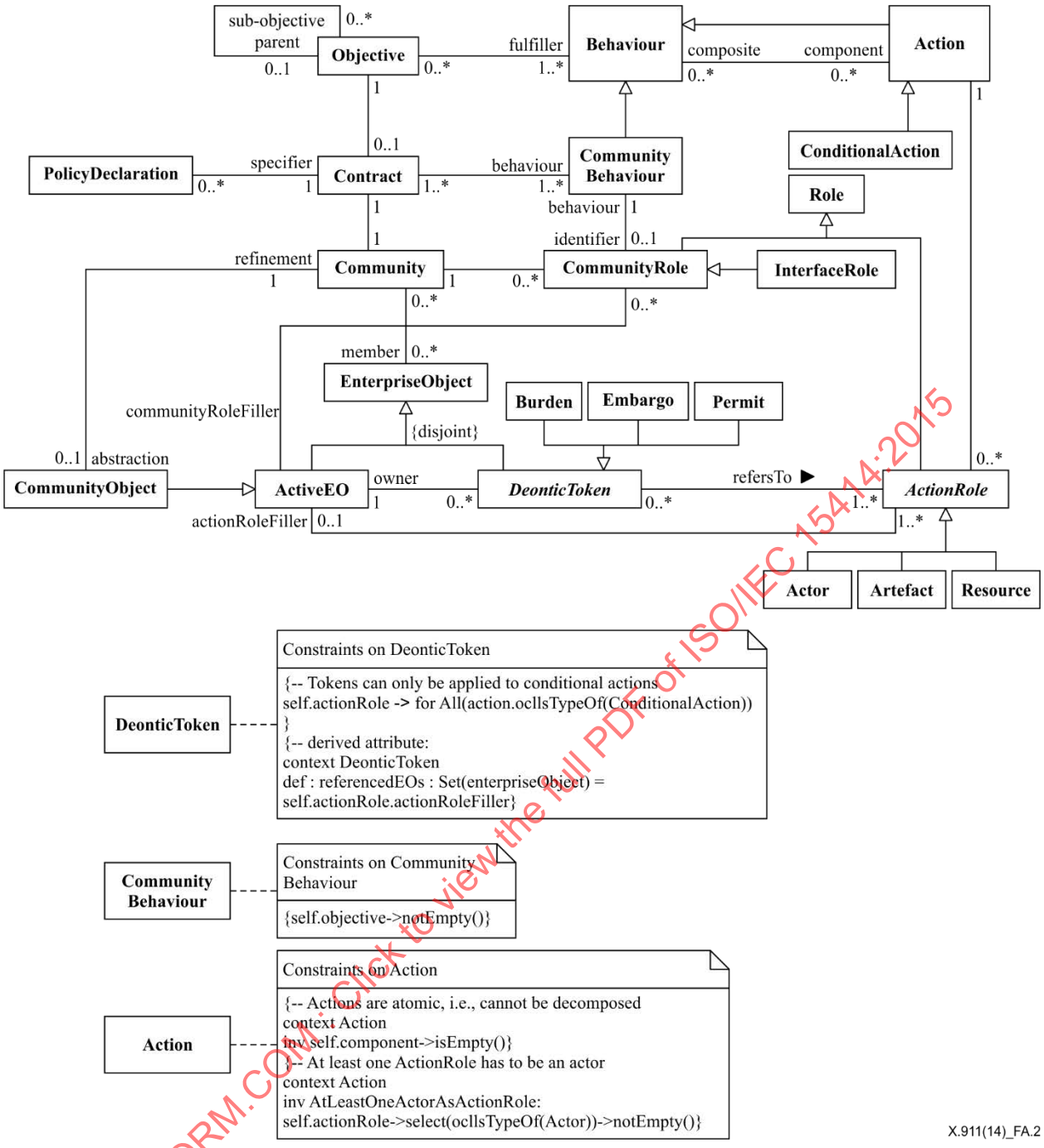

Figure A.2 – Community concepts

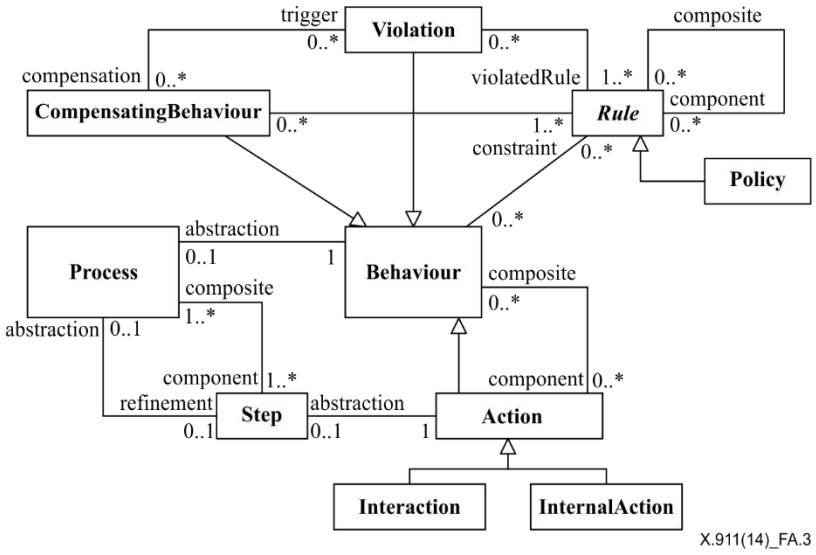

Figure A.3 – Behavioural concepts

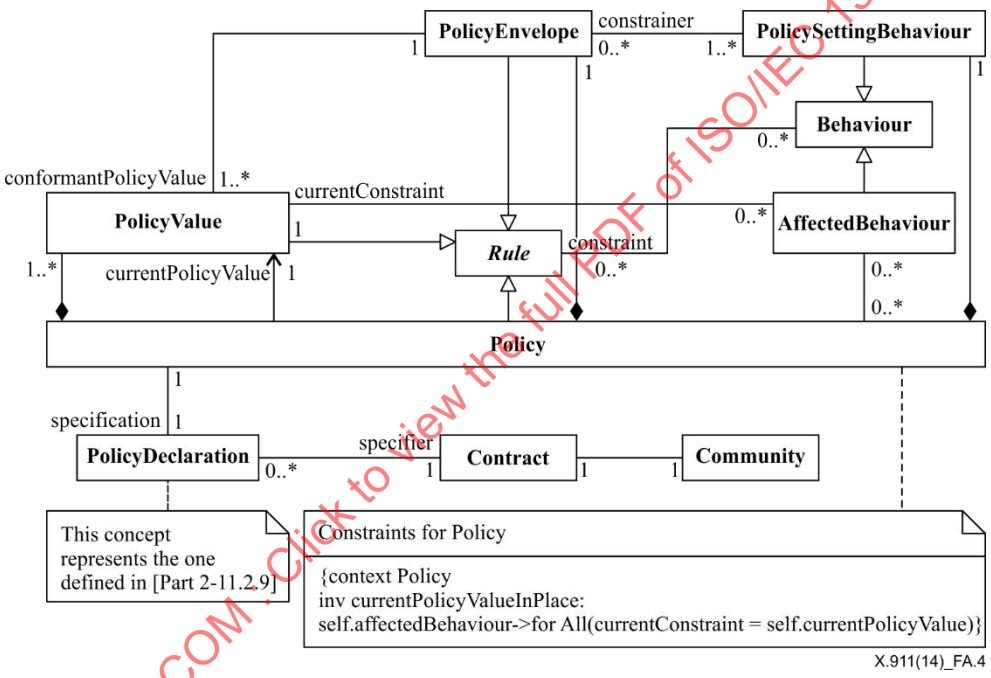

Figure A.4 – Policy concepts

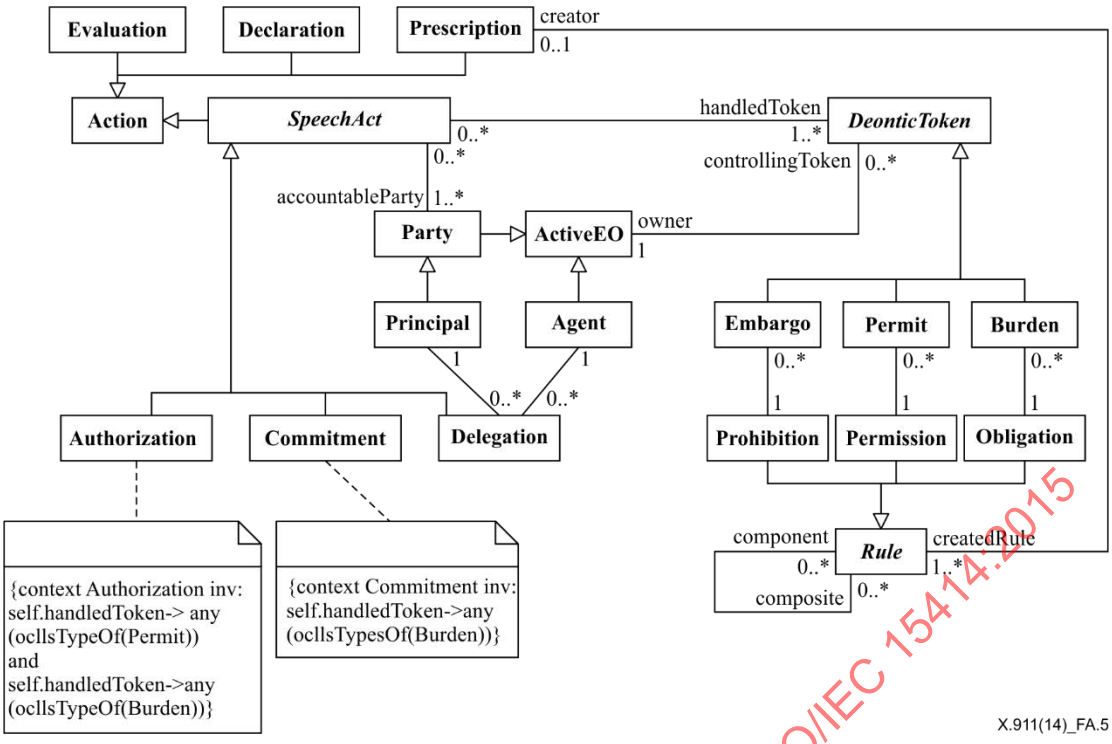

Figure A.5 – Deontic and accountability concepts

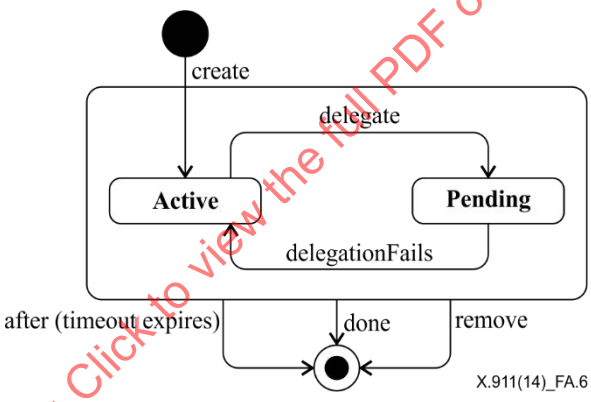

Figure A.6 – Deontic token life cycle

Annex B

Explanations and examples

(This annex does not form an integral part of this Recommendation | International Standard.)

This annex explains the concepts and structuring rules of the enterprise language and provides examples of how they may be used. This annex is informative.

Each of the two parts of this annex includes a running example. The two examples illustrate somewhat different uses of the concepts and structuring rules to specify an ODP system from the *enterprise viewpoint*. [Part 3-4.1]

The concepts explained are:

- Enterprise specification concepts including the *specification* itself, the *field of application* of the specification, the *system*, and its *scope*;
- Community concepts including, *community*, *enterprise object*, *objective*, *contract* of the *community*, life cycle of a *community*, assignment *policy*, relationship between *communities*, *domains* and *federation*;
- Behaviour concepts including *action*, *behaviour*, *role*, *process* and *step*, *interface role* and *violation*;
- Deontic concepts including *authorization*, *obligation*, *permission* and *prohibition*;
- Policy concepts including *policy envelope*, *policy value*, *affected behaviour* and *policy setting behaviour*;
- Accountability concepts including accountability, *party*, *commitment*, *declaration*, *delegation* and *authorization*, *agent* and *principal*, *evaluation*, and *prescription*.

In this annex, *terms* that refer to concepts that belong to the specification language, the RM-ODP *enterprise language*, are in *italic*, *terms* that refer to concepts that belong to the universe of discourse (that is, what is being specified) [Part 2-6], are in the usual roman typeface, and *names* used in an ODP specification are in sans-serif roman. In some cases, a *term*, which refers to a concept that belongs to the enterprise language, is also occasionally used in this annex with its ordinary sense. When such a *term* is used in its ordinary sense, it is in the usual roman typeface, not in *italic*.

B.1 First example – Enterprise specification of an e-commerce system

Our first example is an enterprise specification of an e-commerce system operated by e.com, a seller of widgets.

B.1.1 Specification [Part 3-4.2.2]

A person places an order for a purple widget with the e.com e-commerce system. That person and the e.com e-commerce system are represented in our specification as *objects*. Placing that order for a purple widget is represented as an *action*. Those *objects* participate in that *action*, playing the distinct action roles of customer and supplier.

Several *objects* participate in the *action* fulfilling the order for a purple widget. Each of these *objects* participates in some of the more detailed *actions* in a *decomposition* of the order *action*.

B.1.2 Field of application (of a specification) [6.1.2]

The e-commerce system specification belongs to the domains of trade and electronic business transactions and, therefore, assumes the set of structures and roles common to these domains, such as buyer, seller, order, delivery, item, etc., as well as button, browser, client, server, electronic payments, etc. Moreover, the specification assumes certain ways of doing business and is thus restricted to use only some kinds of valid operations. Thus, the e-commerce system specification can only be applied within the domains of trade and electronic business transactions; that is, it may not make any sense to use this specification in other environments where the aforementioned assumptions are not valid. Examples of such invalid environments include the cases of barter-based trading communities, or systems without computerized resources, or with no access to electronic services (such as e-orders, e-payments, etc.).

B.1.3 System [Part 2-6.5]

The e-commerce *system*, represented by an object, e-system, includes a purchasing *subsystem*, a shipping *subsystem* and an administration *subsystem*. These are represented in the specification by three *systems*, purchasingSubsystem, shippingSubsystem and administrationSubsystem.

Often an *ODP system* will include some parts which are computers, others which are other kinds of machinery, and parts in which people perform some task. The e.com e-commerce system comprises a part that is a computer and parts that are e.com employees. The computer part handles most orders automatically, but refers some for decision-making by the e.com employees.

B.1.4 Scope [6.1.1]

The *scope* of e-system in our e-commerceCommunity includes behaviour providing a list of available kinds of widgets and the price of each, providing representations about the merchantability and fitness for use of each kind of widget, accepting orders and payments, keeping track of inventory, and so on.

B.1.5 Community [Part 3-5.1.1]

The e.com e-commerce system interacts with people and with other automated systems. When the operation of that e-commerce system is specified from the *enterprise viewpoint*, that system and the context in which it operates are represented as a *community*. We will call this the e-commerceCommunity.

In the *enterprise viewpoint* specification of the *ODP system*, the e.com e-commerce system is represented as an *object* in the e-commerceCommunity. That *object*, the e-commerce system object, we call the e-system for short.

People and automated systems are also represented as *objects* in the e-commerceCommunity.

The e-commerce system is composed of parts. To show the *interactions* of the parts, our specification also shows the e-commerceCommunity in more detail, with the e-commerce system shown as decomposed into several *objects* representing its parts.

The *enterprise specification* may include other *communities*. The *objects* of these *communities* will be either the parts of the e-system or other *objects* in the *environment* of the e-system.

In our specification there are also other, larger *communities* that include the e-system or some of its parts. Each of these *communities* has its own *objective*. We will see examples later in this annex.

B.1.5.1 Enterprise object [Part 3-4.2.2]

The main enterprise objects in the e.com enterprise specification are customer, supplier and widget.

B.1.5.2 Objective [6.2.1]

People, firms and automated systems interact with our example system in order to exchange goods and money. The *objective* of the e-commerceCommunity is to enable this exchange. (This *objective* may, of course, be specified in more detail.) This is captured in our *enterprise specification* as a preference that, in the future, goods will have been exchanged and everyone will be satisfied with the exchanges.

B.1.5.3 Contract [Part 2-11.2.1]

The *contract* of our e-commerceCommunity includes *rules* and *policies* about privacy of customers, representations about the goods, placing of orders, means of payment, procedures in case of dissatisfaction, and so on. The *policies* about means of payment will prescribe what methods of payment are permitted, how information about a method of payment will be transmitted, and the like.

This *contract* of the *community* also refers to a legal agreement between e.com and its customers. Some of the provisions of that agreement are represented in our specification as a *policy* that brings together a set of *rules* for the purpose of ensuring that the e-commerce system complies with that legal agreement even if it is changed.

B.1.5.4 Role [Part 2-9.1.7]

The *roles* of our e-commerceCommunity include *roles* for *objects* representing the e-commerce system, customers, widget suppliers, supplier systems, and e.com managers which are *enterprise objects* in the e-commerceCommunity. The *roles* of e-system in our e-commerceCommunity include catalogueServer and orderTaker. Our specification states that the *role* catalogueServer has behaviour which includes displaying a welcome page, displaying catalogue pages, searching for kinds of widgets that meet a need described by a customer, and so on. This is just one of the *roles* of e-system in our e-commerceCommunity.

Our specification includes the *roles*, customer and e.comManager. Because e.com is pleased to have its employees as customers, the same object, representing a manager of e.com, may fulfil both the *roles* of customer and e.comManager.

Another *role* in our specification is the *role*, auditor. As a matter of company policy, employees acting as auditors are not allowed to use the system as customers. So, our specification states that an *object* fulfilling the *role*, auditor, may not fulfil the role customer.

We can use a shorthand such that customer object means an *object* fulfilling the *role*, customer.

In the *action* of a customer buying a purple widget, e-system *object* and the customer *object* are *actors*, the *object* representing a purple widget is an *artefact*, and shippingSubsystem is both an *actor* and a *resource* (represented in the *action* by a *community object* (a *composition* of the parts of that subsystem), which will be in charge of delivering the product at a later stage).

B.1.5.5 Interface Role [6.3.5 and 7.8.3]

In the specification, e-system is a *composite object*. Some of the *components* of e-system fulfil *roles* in the inventoryMaintenance *community*. This *community* interacts with supplierSystem *objects* (*objects* fulfilling the *role* supplierSystem) outside the inventoryMaintenance *community*. The inventoryMaintenance *community* includes *interface roles*. The *objects* of this *community* that interact with supplierSystem *objects* fulfil *interface roles*.

B.1.5.6 Establishing a community [7.6.1]

The specification of our e-commerceCommunity includes behaviour to establish a community of type, justInTimeCommunity, with a supplier *object* representing a widget supplier who agrees to maintain an inventory of widgets sold by e.com, and deliver them to e.com warehouses as directed by the inventoryMaintenance *object*. When a supplier agrees to do so, a new *community* is established, comprising the supplier *object* and the inventoryMaintenance *object*.

B.1.5.7 Assignment policy [7.6.2]

The *assignment policy* of our e-commerceCommunity includes a *rule* providing that for an *object* to fulfil the customer *role* it shall be an *authenticated object*. (Our specification includes the specification of a securitySubsystem, which provides an *authentication function* for *objects* connecting to the e-commerce system.)

The specification includes *rules* for assigning *parties* to the *role* widgetSupplier. When e.com agrees to purchase from a new supplier, a new *party object* is *created* in e-system to represent that supplier and that *party object* fulfils the *role* widgetSupplier. The specification includes *rules* for *introducing* new *objects*. When e.com agrees to purchase from a new supplier, an object representing that supplier's system is introduced and fulfils the role supplierSystem.

The specification includes *rules* for assigning existing employee objects to the role manager.

B.1.5.8 Relationship between communities [7.3.2, 7.8.3]

e-system is a *configuration of objects*, including a purchasing *subsystem*, a shipping *subsystem*, and an administration *subsystem*. The *enterprise specification* includes a *community*, e-systemCommunity; the *objects* of that *community interact* to realize the *objective* of that *community*. e-system is a *composite object*, a *composition* of the *objects* of e-systemCommunity. The *enterprise specification* also includes supplyCommunity. The *objects* of that *community* are e-system *objects* representing the sales systems of companies that supply e.com, and *objects* representing systems of other companies that are also customers of the supplying companies. When e-system participates in supplyCommunity, this is an *interaction* of supplyCommunity and e-systemCommunity.

The *enterprise specification* includes a *community* corresponding to each of the *subsystems* of e-system, each with its own *objective*, *policies*, etc. So there are purchasingCommunity, shippingCommunity, warehouseCommunity, and accountingCommunity. Each of these *subsystems* is also a *configuration of objects* (the parts of that *subsystem*); each of these *objects* fulfils one or more *roles* in the corresponding *community*. When an *object* acting in a *role* in warehouseCommunity, namely the inventoryMaintenance *object*, interacts with an *object* fulfilling the supplyPlanning *role* in the purchasingCommunity, by providing information for use in purchasing widgets to restock inventory, this is an *interaction* between warehouseCommunity and purchasingCommunity.

When an *object* is acting in the *role* inventoryMaintenance in the warehouseCommunity, and that *object* also fulfils the *role* assetReporter in accountingCommunity by providing information for use in daily asset accounting, which that object obtains while fulfilling the role inventoryMaintenance, this is an *interaction* between warehouseCommunity and accountingCommunity.

The e-system may contain an *object* dedicated to monitor the business activities performed in e-commerceCommunity. This *object* may process the data it gathers and produce reports of preferred supplier ranking and active buyer ranking. There is also a ratingServiceCommunity. These two *communities* are operated independently, but exchange information. For that purpose, e-commerceCommunity specifies an *interface role*, fulfilled by the *object* monitor that provides local ranking information and receives nation-wide ranking information. The ratingServiceCommunity specifies an *interface role*, fulfilled by an *object* in that *community*, which receives company-specific ranking data and provides nation-wide ranking information. These two *interface roles* enable *interaction* between the *communities*.

The e-system (a *community object*) is subject to the *policies* of supplyCommunity.

In addition to being subject to their own individual policies, all the community objects representing the different subsystems of e-system (purchasingCommunity, shippingCommunity, warehouseCommunity, etc.), are subject to the policies of e-system.

The *object* fulfilling both of the *roles* inventoryMaintenance and assetReporting, is subject to the *policies* of two different *communities* at the same time, warehouseCommunity and accountingCommunity. Other *objects* in warehouseCommunity may be subject only to the *policies* of warehouseCommunity.

The *object* monitor is subject to the *policies* of e-commerceCommunity and to the *policies* of the information exchange *community* composed of monitor and the *object* in the *interface role* of ratingServiceCommunity.

In addition to a closed business-to-business supply chain community, the e-commerce system may make use of the web services infrastructure, by registering its services to an open registry, in order to find new potential customers. And, when the need for new customers is satisfied, the e-commerce system may unregister the services from the open registry. Registration with the open registry is represented as *establishing behaviour*, and the open registry, the e-commerce system, and the potential customers using the registered web services are represented as a newly created *community*.

B.1.5.9 Domain [Part 2-10.3]

The parts of the e-commerce system are all under control of e.com. Other *domains* in that *system* include:

- a security *domain*, comprising *objects* providing data access and *communication* services, subject to *policies* set by a security authority *object*, securitySubsystem.
- a naming domain, with named objects, which obtain their names from a naming service object.
- an audited *domain*, comprised of *objects* which are audited by a certain auditor *object*.

B.1.5.10 Federation [Part 3-5.1.2]

The automated systems in our example are not all under common control. The e-commerce system is under control of e.com. It interacts with, for example, another automated system controlled by a customer. We have many *administrative domains* in our specification; these include the *domain* of automated systems controlled by e.com, as well as a domain controlled by each customer. The e-commerce system also interacts with people; each person is, ultimately, controlled only by herself or himself. (Although, we might, if it is useful in our specification, consider *objects* representing employees to be members of *domains* controlled by *objects* representing their employers.)

The *objects* of e-commerceCommunity are not all under common control. The supplier systems are under administrative control of each of the suppliers. There are several immediately visible "domains", for example, the e.com domain and the customer IT system domain. Within the e-system itself, there are also domains under separate control. All objects in the e-system are in a single securityDomain, with *characterizing relationship*, subjectToSecurityPolicySetBy and *controlling object* securitySubsystem. But the objects of purchasingSubsystem and shippingSubsystem are in a policy *domain*, with *characterizing relationship*, setsPoliciesFor, with *controlling object* fulfillmentDivisionExecutive, while the objects of securitySubsystem are in a different policy domain, with *controlling object* CIO.

An *enterprise viewpoint* specification might specify a number of separate federations, such as document management federations, billing management federations, customer management federations, and so on. Or it might take a unified approach, in which all these aspects are captured into a single federation community. This federation community might also include a mechanism for incorporating additional functionality, to achieve a new, shared objective.

B.1.6 Behaviour [Part 2-8.7]

Behaviours of the e-commerceCommunity include placing an order, shipping a widget, and collecting a payment; these all involve several *interactions* between e-system and *objects* in the environment. Internal *behaviours* include keeping track of inventory and making decisions about replenishing inventory.

B.1.6.1 Action [Part 2-8.3]

Actions in the e-commerceCommunity include *internal actions* of e-system, such as changing an inventory record and comparing the count of an item in inventory to the reorder limit for that item, and *interactions* between e-system and *objects* in its *environment*, such as requesting a price, adding a line item to an order, or requesting a payment.

B.1.6.2 Process [6.3.6]

The specification of the inventoryMaintenance *community* includes the specification of a reorderProcess. This *process* includes the *objects* orderPlacer, receiver and inventoryKeeper and the *role* supplier. One *step* in this *process* is the *action* in which the orderPlacer places an order. The *process* of placing an order proceeds in the same way for any supplierSystem fulfilling the role supplier.

For handling goods as they are received, the specification of the inventoryMaintenance *community* includes the specification of a receivingProcess. This *process* includes a *forking action*. Following the fork, in one *chain* the inventoryKeeper adjusts the *inventory*; in the other *chain*, the orderPlacer adjusts outstanding *orders*. The *steps* of these two *chains* are followed by a *joining action*; this is followed by *steps* completing the receivingProcess.

B.1.6.3 Violation [6.3.8 and 7.8.6]

An *action* of securitySubsystem, which prevents an auditor object from examining a certain record, is a *violation* of the *rule*, mentioned in B.1.7.2 below, that a *party* in the role of auditor is *authorized* to examine any record in the e-commerce

system. An *action* by a program executed by a databaseAdministrator *object*, which displays an employee salary, is a *violation* of the *rule*, mentioned in B.1.7.5 below, that prohibits display of a salary record except for *objects* fulfilling certain *roles*.

When an order is accepted before 4 PM, but ordered widgets, which are in stock and not already scheduled for shipment at the time of acceptance of that order, are not scheduled for shipment the same day, this behaviour by the shippingSubsystem is a *violation* of the *rule*, mentioned in 0B.1.7.3 below, that obliges shippingSubsystem to schedule such shipments on the same day.

B.1.7 Deontic concepts

B.1.7.1 Deontic tokens

Many of the examples introduced above involve actions that are subject to permissions, prohibitions or obligations. For example the orderWidget action involves both permissions and obligations; it is therefore specified to be a speech act and the e-system *object* needs to hold a *permit* deontic token before it can participate in the action in the *role* of orderTaker. Performance of the action results in creation of a *burden* deontic token expressing the obligation to deliver a purple widget. Initially this is held by the orderTaker, but it may be transferred as part of a subsequent delegation to the shippingSubsystem. Another *burden*, expressing the obligation to make payment, is created as held by the customer *object*.

B.1.7.2 Authorization [6.6, 7.8.8.4]

The *enterprise specification* of the e-commerce system includes a *rule*, which prescribes that an auditor *object* (that is, a *party* fulfilling the *role*, auditor) is *authorized* to examine any record in the system; this is modelled by the auditor holding a *permit* deontic token. Another *rule* prescribes that a databaseAdministrator *object* is *authorized* to display records when testing the operation of the dataManagementSubsystem; this is also modelled in terms of a *permit* deontic token.

B.1.7.3 Obligation [Part 2-11.2.4]

The *enterprise specification* includes a *rule* which prescribes, for any order accepted before 4 PM, to schedule shipment on the same day of all ordered widgets in stock and not already scheduled for shipment at the time of acceptance of that order. This *rule* is represented as an *obligation* of the shippingSubsystem, in terms of an associated *burden* deontic token, created by the orderWidget *speech act* and then assigned by delegation to the shippingSubsystem.

B.1.7.4 Permission [Part 2-11.2.5]

The *enterprise specification* includes *rules* that certain *subsystems* may establish *communication* with *objects* outside the e.com *security domain*. These *rules* are *permissions* and are represented as *permit* deontic tokens.

The *enterprise specification* provides that *objects* in the *role* orderPlacer, have *permission* to *establish communication* with *objects* outside the e.com *security domain*. An orderPlacer *object* attempts *communication* with an *object* representing a particular supplier system. securitySubsystem prevents that *communication* because of a temporary specific *prohibition*, represented as an *embargo* deontic token, on *communication* with that supplierSystem *object,* because it currently appears to be a zombie in a denial of service attack on e.com. This is not a *violation* of the *permission* of that orderPlacer *object*.

B.1.7.5 Prohibition [Part 2-11.2.6]

The *enterprise specification* of the e-commerce system includes a *rule* that forbids an employee from assigning a new party to the role widgetSupplier while that employee is being investigated for an irregularity by an auditor. This rule is specified by the passing of an *embargo* deontic token for the auditor to the employee affecting the *action type* assignSupplier.

In many cases, the default is for there to be a prohibition, which is overridden where necessary by an explicit permission. For example, a *rule* prescribes that a salary record may be displayed only for a salary administrator, an auditor, or a manager of that employee. Another *rule* prescribes that no *subsystem* may *establish communication* with a *system* outside the e-com *administrative domain* without a *permission* included in that specification or granted by securitySubsystem. Any such *communication* in the absence of a *permission* is a *violation*.

B.1.8 Policy [Part 2-11.2.8, 6.5]

The specification of the e-commerce system identified a *policy* governing the treatment of back-orders; it defines what the system should do if the supplier cannot fill the entire order. The specification includes *a policy declaration* that provides a number of pieces of information about the policy. The *affected behaviour* is the termination of order processing when not all items are available. The *policy envelope* is the set of all functions that yield a binary decision about whether or not to cancel the order, based solely on information about the supplier. The *default policy* value is a function to allow back-order when dealing with favoured suppliers and to cancel the order otherwise. The *policy setting behaviour* is the

performance of the setBackOrderPolicy action by the holder of the shippingSystemManager *role* in the shippingSystem *community*.

B.1.9 Accountability [6.6 and 7.10]

If a customer makes a purchase from an e-commerce system, this is represented in the *enterprise specification* as an *action* of the *object* representing that customer.

If the purchasing agent of e.com makes a purchase (in her capacity as purchasing agent), this may be represented as an *action* of the *object* representing e.com, an action of the *object* representing the person (the purchasing agent), or both. The choice depends on the purposes of the specifier or owner of the system.

If a firm's computer system sends an order to the e-commerce system (and has a permit giving delegated authorization to do so) this is represented as an *action* for which the *object* representing that firm is *accountable*.

B.1.9.1 Party [6.6.1 and 7.10.1]

The *objects* representing people and the *objects* representing e.com and other firms are *parties*.

B.1.9.2 Commitment [6.6.2 and 7.10.3]

A firm may operate a computer system which sends an order to the e-commerce system. The sending of the order by that computer system commits that firm to pay for the goods when timely delivered. (For example, e.com may have a contract with that firm which provides that, or this may the result of the operation of commercial law.) The sending of that order is represented as a *commitment*.

B.1.9.3 Declaration [6.6.5 and 7.10.4]

e.com has agreements with its customers which provide that, when an order is cancelled within twenty-four hours of the scheduled shipping data, e.com has the option to make a restocking charge of 5% of the invoice amount for the cancelled order. The e-commerce system is programmed to make that charge automatically in the case of a customer that has accounts receivable by e.com which are more than sixty days late in payment. The cancellation of an order by a firm, F, is represented in the specification as an *action*, cancel, by an *object*, firm F, referring to another *object*, order. An *action* of e-system invoking the restocking charge when an order is cancelled by firm F is a *declaration*. That *action* establishes a state of affairs in the *environment* of e-system: firm F is obliged to pay that charge to e.com.

B.1.9.4 Delegation and authorization [6.6.4, 6.6.6, 7.10.1 and 7.10.2]

The *specification* provides that e-system may be *delegated* the authorization to cancel a contract with a supplier or customer of e.com (subject to the terms of that contract). Sending a message to the counterpart cancelling a contract is represented as a *declaration*: the *communication* of that message causes the contract to be cancelled. This will be the case, for example, when the e-commerce system automatically cancels an order.

B.1.9.5 Agent and principal [6.6.8, 6.6.9 and 7.10]

In our *enterprise specification*, e.com is represented as an object, e.com. The person acting as chief financial officer (CFO) of e.com causes the e-commerce system to serve offered prices to the computer systems of customers (including web browsers and purchasing systems), which prices are determined by the CFO and entered into the e-commerce system.

This setting of offered prices is represented as an *action* of the *party*, CFO, representing the CFO. The CFO may delegate to the e-commerce system the authorization to set prices during evenings and weekends. The CFO may also delegate authorization to that system to further *delegate* this authorization to an independent but federated pricing service in certain cases. e-system is the *agent* of CFO and CFO is the *principal* of e-system. If delegated by e-system pursuant to that authorization, pricingServiceP (representing a federated pricing service) becomes the *agent* of CFO and the CFO is the *principal* of pricingServiceP. This is represented by CFO cloning the relevant *permits*, *embargos* and *burdens*, and transferring them to pricingServiceP as part of the delegation *speech act*.

In a different *enterprise specification*, it may instead be e.com that *delegates* to e-system the authorization to set prices during evenings and weekends. e.com may also *delegate* authorization to the e-system to further *delegate* this authorization to pricingServiceP. e-system is the *agent* of e.com and e.com is the *principal* of e-system. If that authority is delegated by e-system to pricingServiceP, then pricingServiceP is the *agent* of e.com and e.com is the *principal* of pricingServiceP.

Alternatively, another *ODP system* provides an e-commerce service, which an application service provider offers to many firms. Each firm using that system is represented in the specification of that system by an object in the role firm.

The *enterprise specification* of that e-commerce service provides a means for firm to *delegate* the authorization to set prices. If e.com uses that service, during the operation of that system, the object, e.com, fulfilling the role, firm, may

delegate the authorization to set prices to its pricingManager (a *role* for an *object* representing the person in the firm with authorization to set prices). In the matter of setting prices, pricingManager is the *agent* of e.com.

If pricingService changes an offeringPrice, this *action* is a model of something that happens in the world of e-commerce. What happens is that there is a new state of affairs: e.com is now offering to sell at the changed price.

In a fully automatic e-commerce system, the *enterprise specification* will provide that the posting of a changed price *object* by an *agent* of e.com (for example, the pricingService) constitutes an offer by e.com to sell at the price indicated by that price. In that case, the effect of the *action* by the pricingService to change that price is this: e.com has made an offer and is committed to sell at the price indicated by price to any customer accepting the offer before it is withdrawn. (This is the representation in the specification of the situation in the world, and exactly represents that situation: The effect of the posting by the federated pricing service of the changed price is this: e.com has made an offer and is committed to sell at the price indicated by the posted price to any customer accepting the offer before it is withdrawn.)

This offer (the posting of the changed price *object*) is an *action* of an *agent* of e.com, for which e.com is accountable. In this case, the *action* is a *commitment*, as e.com is obligated to sell at that price if the offer is accepted.

Since the e-commerce system is a fully automatic system operating over the Internet, an acceptance of an offer to sell will be transmitted to the e-commerce system by software connected to the Internet, perhaps a web browser. In an *enterprise specification* that includes customers and their systems, a webBrowser is the *agent* of an *object* in the *role*, customer and acts for the customer (the web browser acts for the customer by sending the order message when the customer clicks the Buy button).

The parts of an *ODP system* may be specified in this same way. Within the pricingService, the *object*, priceSelector, may *delegate* current market analysis to one or another marketAnalysisSubsystem depending on the *type* of *product*.

B.1.9.6 Evaluation [6.6.7 and 7.10]

In order to replenish inventory, the e-commerce system prepares requests for bids and transmits them to widget suppliers. When bids are received, the e-com purchasing agent determines which bid or bids to accept. The decision considers estimation of the value of each bid, which the e-commerce system prepares. The e-commerce system assigns a relative status to each bid, according to an algorithm that produces an estimate of the value of that bid, using not only the price, but also the delivery terms offered, records of the previous on-time performance of that supplier, and records of receiving inspection reports on the quality of widgets from that supplier. This assignment of status is an *evaluation* by e-system of the bids.

Other examples of automated *evaluation* are credit scores and insurance underwriting ratings.

B.1.9.7 Prescription [6.6.3 and 7.10.5]

The CFO of e.com from time to time sets *rules* of the *policy* governing the granting of credit by the e-commerce system to established customers. The e-commerce system also includes a credit history evaluation *subsystem*, which is *delegated* authorization by the CFO to change some rules of the credit policy. Such *actions* by CFO and creditHistoryEvaluationSubsystem are *prescriptions*.

B.2 Second example – Specification of a library

This clause provides a second example to illustrate the use of the *enterprise language* concepts and structuring rules to specify an *ODP system*. The example describes the elements that comprise an enterprise specification of a library. Thus, in this case, the *ODP system* is a business system (which may or may not include a computer system). The example shows how RM-ODP specifications (in particular, those from the *enterprise viewpoint*) can be used to specify systems other than computer systems.

This example is based on a university library, especially on the regulations that rule the process of borrowing items from that library. Instead of a general and abstract *system*, this example is loosely based on the regulations defined for the Templeman Library at the University of Kent at Canterbury, a *system* that has been previously used by different authors for illustrating some of the ODP concepts. The rules that govern the borrowing process of that library *system* are as follows:

- Borrowing rights are given to all academic staff, and to postgraduate and undergraduate students of the university.
- There are prescribed periods of loan and limits on the number of items allowed on loan to a borrower at any one time. These limits are detailed below.
- Undergraduates may borrow eight books. They may not borrow periodicals. Books may be borrowed for four weeks.

- Postgraduates may borrow 16 books or periodicals. Periodicals may be borrowed for one week. Books may be borrowed for one month.
- Teaching staff may borrow 24 books or periodicals. Periodicals may be borrowed for one week. Books may be borrowed for up to one year.
- Items borrowed shall be returned by the due date and time.
- Borrowers who fail to return an item when it is due will become liable to a charge at the rates prescribed until the book or periodical is returned to the library.
- Failure to pay charges may result in suspension by the Librarian of borrowing facilities.

NOTE – Unless otherwise stated, all references in this clause to the Library community will refer to the *community* representing the Templeman Library and its environment, whose borrowing regulations have been described above. Other *communities* will also be mentioned in this annex, such as the University community that represents the university that the library serves, and a banking community representing the use by the library of services offered by banks. Some variations of the Templeman Library *community* will also be introduced for illustration purposes. For instance, for illustrating the *accountability concepts* of the enterprise viewpoint language, we will introduce another library that uses a computerized system to keep track of the borrowers and their outstanding loans.

B.2.1 Enterprise specification

Four key concepts of *enterprise language* are: *system*, *scope*, *enterprise specification*, and *field of application*. The following points in this clause (B.2.1) describe these four concepts. Clauses B.2.2 to B.2.3 focus on the elements that constitute the *enterprise specification* of the *system*: *communities* and *behaviour*. Clause B.2.4 introduces *deontic concepts* and clause B.2.5 deals with *policies*. Clause B.2.6 concentrates on the *accountability* concepts.

B.2.1.1 System

The *system* we want to specify is a university library, in particular (a reduced and simplified version of) the Templeman Library at the University of Kent, Canterbury, whose borrowing regulations have been given above. This *system* (hereafter called the 'Library System') and its *scope* are described by an *enterprise specification*.

B.2.1.2 Scope [6.1.1]

The *scope* of the library system describes its expected *behaviour*, i.e., the way it is supposed to work, in terms of *roles* (see clause B.2.2.4) or *processes* (B.2.3.2) or both, *policies* (B.2.5), and the relationships of these. All these elements will be described below.

B.2.1.3 Enterprise specification [Part 3-4.2.2]

The library system and the environment in which it operates are represented as one *community*, the libraryCommunity. The *enterprise specification* will state the *objective* of that *community*, how it is structured, what it does, and what *objects* comprise it.

B.2.1.4 Field of application

The library in the example is the library of a university, and therefore assumes the existence of some structures and roles typical of those organizations. It may not make sense if we try to use this specification for a library in the Army, where no such concepts as academic staff or students can be easily applied.

B.2.2 Community

B.2.2.1 Community [Part 3-5.1.1]

In the specification of the library system, the library is represented as a *community*; the behaviour of that community is specified in the following using *roles*, *processes* and *polices*.

B.2.2.2 Objective [6.2.1]

The library system maintains a collection of books, periodicals, and other items, that may be borrowed by its members. The library system comes into being with the establishment of its collection, with the primary *objective* of "sharing this collection amongst the University members".

B.2.2.3 Contract [Part 2-11.2.1]

The *contract* of the library system community specifies the *objective* of that *community* and how this *objective* can be met. That *contract* specifies the different *roles* that *objects* may fulfil in the *community* (i.e., its structure and behaviour) and the *policies* that govern the *behaviour* of these *objects* while fulfilling *roles* in the *community*.