Finland votes "yes" for the progression of Part 3 to Draft International Standard, with the following comments:

The main areas of concern in this commentary are: Firstly, concept of binding, especially the level of detail in its introduction in computational viewpoint language, and secondly, concepts related to transparencies and function. The second group covers introduction of transparencies, and introduction of functions, especially introduction of management functions, as well as relationship between transparencies and functions.

Minor clarifications required are related to concepts of federation, community and domain, and interceptors, also.

A. Binding

Item: A1
Clause: 17 Computational language
Importance: Technical
Rationale:

> In computational viewpoint only the functional decomposition of an ODP system is under consideration. That means, that logical, interesting objects are shown. As well, their configurations and logical operations needed to fulfill their mission are shown.
>
> Communication between these logical objects can be viewed as interrogations or announcements that have communication quality properties, representing contracts between the mission objects and the supporting system. This means that a liaison (or binding) between objects have attributes. For computational viewpoint considerations this is enough. Look Figure A.1.
>
> In order to explain, how the attributes or properties of the communication are implemented, a whole subsystem is introduced. Look Figure A.2.
>
> The RM−ODP computational language should have tools for description of this subsystem also. However, the description of the communication subsystem is not part of the RM−ODP framework (i.e., Part 3), but the Binding component standard (the NWI suggested in Southampton).

Proposed improvement:

Move concepts of binding object, binding object template, signal binding object template, stream binding object template, and operational binding object template from RM−ODP Part 3 to the new binding issue.

Also, move clause 7.2.1. similarly.

Review the interaction rules, type rules, template rules, portability rules, and failure rules to correspond the text still existing. Extra material should be again fed in to the NWI.

Item: A2
Clause: ???
Importance: Technical
Rationale:

An object that is instantiated within a node gets a binding with the nucleus and also with the trader object. It should be enough to be bound initially to either of those. It is more useful to be bound to the nucleus first. In that way the object may request the nucleus where it should be bound for trading services.

Proposed improvement:

Delete the sentence.

B. Management functions

Item: B1
Clause: 12
Importance: Technical, editorial
Rationale:

RM−ODP describes a general infrastructure that supports distribution transparency of services within a coalesced network of autonomous domains. This infrastructure is strongly based on concepts of nodes, capsules, clusters and their manager objects. The draft standard defines node, capsule and cluster manager objects together with a set of operations, but the overall interworking schema is left unclear. Although, it is not necessary to describe the ODP infrastructure in a detailed way within a prescribing framework standard like RM−ODP Part 3, we feel that a more coherent picture of cluster, capsule and node management functions should be given.

Clusters, capsules and nodes, as described in the current RM−ODP document, seem to lack almost all aspects of traditional management schemata and exhibit only features of typical operating system services. The traditional OSI management schema requires that there are two parties involved in each management operation: a managing system and a managed system. In some cases the operations of manager objects of ODP infrastructure seem to belong to the managed object part, sometimes to the managing object part. The description could benefit from separating these two roles in each object involved and thereby clarifying the collective behavior.

In the current draft standard, the management functions are identified to be used towards objects identified in the computational viewpoint. But, unfortunately, also the description of these functions cover only the computational viewpoint specification. According to the RM−ODP specification rules, every subsystem should be specified using all five viewpoints. Object management is a general function that can be in particular applied to objects within the infrastructure, e.g. nodes, capsules, and basic engineering objects. It can also be applied to general application management.

The management of object and their configurations will be further specified in ODP management framework. For this work, the RM−ODP framework and infrastructure should give clear advise.

For the base of management functions introduction, concepts of object and role should be taken. When concerned with the management of objects, it is very natural to see the objects in at least two roles. Every object has its normal mission role, i.e. the purpose in was designed for. Besides this, it has a management role, the behavior needed to monitor and control the mission role behavior. Both roles have their own interfaces, the mission role offers a mission interface and the management role offers a management interface. As management function is by nature asymmetric there are two interworking roles involved in management, a managing role and a managed role. The management interface of an object in the managing role is bound to the management interface of an object in managed role.

An object can have a managed object role within one community, and at the same time, a managing object role towards an object within another community. This kind of object could be called a controller, to stress its dual management oriented roles. For example, the controllers mission might be the managing of an object applying rules given by an administrator or upper layer controller.

Proposed Improvement:

In the introduction of node, capsule and cluster management show also the enterprise and information viewpoints of management functions:

Management functions purpose from the enterprise viewpoint is to control the behavior of a managed object. Control over the managed object is run by another object, a managing object. A managing object may set restriction to the managed object's behavior, or request what state the managed object has. Roles of managing object and managed object within different communities may coexist in one object. In that case, the object is called a controller.

Management related information covers two areas: Firstly, there is a set of behavior restricting rules (policies) describing the state and possible state changes of an object. This information is the effect taking part within the managed object that is only part of the more global policy regulating the activities of the whole system. For the object to respect policies of the community it belongs to, it must privately store part of that information as rules, and use those rules to guide its own behavior at all its operations. Secondly, the managed object has some internal state related to its mission. The environment of the object also included into the same system, may need information about that internal state. Thus the environment needs means to retrieve that information from a managed object.

C. Transparencies and functions

Item: C1
Clause: 14 and 16
Importance: Technical, editorial
Rationale:

On the editorial side, this item suggests a larger reorganization of presented material. The reorganization would clarify the goals of ODP framework, make the Part 3 description more readable and would provide means to correlate the descriptions of functions and the descriptions of transparencies. The reorganization is basically done by changing the order of clauses 11−15 and 16. In this way, transparencies are described first and they can be easily used as rationale for the functions.

On the technical side, this item suggests that the now implicit correspondence between transparencies and functions were written in explicit form. Also, this starting point would lead to more natural grouping of functions introduced. The RM ODP should identify a framework of functions that all open systems must offer. Now, there seems to be some functions included that are useful, but do not necessarily have the position of being compulsory for all open systems. Relationship from transparencies to functions helps in identifying the essential functions.

All the statements made in the following "proposed improvement" text are collected from various clauses already in the Part 3. So, no new interpretations or additional material, except the figures, are suggested. The suggestion only regroups descriptions and gives new titles.

The fundamental grouping rule in functions have been to view to groups:

- functions supporting transparency
- "mission functions" that support services relevant for any computing system, for example security.

Proposed improvement:

Use the following outline for chapters 11–16.

11. ODP transparencies

- define /refer to definition of ODP transparency, i.e., distribution transparency
- "distribution transparency relates requirements expressed in the enterprise, information and computational languages to structures in the engineering language"
- include current clause 16 as whole, except those rules at every transparency description referring to functions
- this will leave some transparency clauses totally empty: they should be populated with the purpose of that transparency in enterprise viewpoint terms
- possibly use a figure like Figure C.1: Distribution transparency classes

12. ODP functions

Quotes from different clauses of Part 3:

- functions are required to achieve open distributed processing
- functions are fundamental to construction of ODP systems based on the computational and engineering languages
- functions can be combined to form specializations for components of ODP systems (component composition standards)
- objects used to model functions do no define necessary structure in implementation

Specially:

- functions are required to support distribution transparency
- include clause 11 as whole
- figure 7 in clause 11 is incorrect and should be deleted (see item C2)
- include from current clause 16 Transparencies all information related to correspondence between each transparency and a set of functions
- include figure shown in Figure 2 (see item C3)
- this method only identifies the functions needed to support

transparencies

- another group of functions might be generated from the essential mission of any, general open infrastructure (for example, such functions that support time concepts)

13. Management functions

- include current clause 12 as whole (see also item C6)

14. Coordination functions

- include current clause 13 as whole (see also item C7)

15. Repository functions

- include current clause 14 as whole (see also item C8)

16. Security functions

- include current clause 15 as whole (see also item C9)

17. "Mission" functions

- (see items C6–C9)

Item: C2
Clause: 11, Figure 7
Importance: Editorial
Rationale:

The figure is very misleading. It introduces functions that do not exist elsewhere in Part 3. It also leaves out some essential functions. Furthermore, it includes loops.

Proposed improvement:

Delete the figure or replace it. A replacement is suggested in item C1, Figure C.2.

Item: C3
Clause: –
Importance: Technical
Rationale:

Figure C.2 of item C1 shows some inconsistencies of the text that should be resolved. The figure has been composed reading the current text and including only the functions or relationships that has been mentioned in the text.

Dotted lines in the figure mean, that the relationship is stated either only in Trading function standard or is implicit, but not clear in Part 3.

Proposed improvement:

Include the following statements on the indicated clauses:

Type A transparency need the support of trading function. All repository functions need the services of storage function. Trading function need the support of type repository function and x (see clause C4).

Name transparency classes A–G with illustrative names.

Item: C4
Clause: –
Importance: Technical
Rationale:

Policy management function was identified as a new base function by the trading function group in the Southampton meeting.

Proposed improvement:

Add the policy management function into the function framework. It might be combined with object management function?

Item: C5
Clause: 15
Importance: Technical
Rationale:

Security functions are not connected to any transparency or other function. Security framework is therefore left in isolation.

Proposed improvement:

Look item C1.

Item: C6
Clause: 12. Management functions
Importance: Technical
Rationale:

In some cases, the term management function is not appropriate. There seems to be two distinct sets of functions. The first set is suitable for this clause and can be called management functions. The second set fulfills more the missions of any open computing platform. These functions should be renamed, for example, using terms like control or service. This set of services should be moved in the new clause 17. "Mission" functions.

Proposed improvement:

Move functions

- clock access and timer management

- thread management
- channel creation
- engineering interface reference management function

to the new clause 17.

Item: C7
Clause: 13. Coordination functions
Importance: Technical
Rationale:
Proposed improvement:

Move

- event notification function
- migration function
- acid transaction function

to the new clause 17.

Item: C8
Clause: 14. Repository functions
Importance: Technical
Deleted.

Item: C9
Clause: 15. Security functions
Importance: Technical
Rationale:
Proposed improvement:

Move all security functions to new clause 17.

Item: C10
Clause: All clauses about functions
Importance: Technical
Rationale:

Following the RM ODP modeling rules.

Proposed improvement:

Each function should be described not only a computational objects, but from enterprise, information and computational viewpoints. An example for this can be found in the article attached to this ballot.

Item: C11
Clause: 16.7. Replication transparency
Importance:
Rationale:

Replication transparency is not included in any class of distribution transparency.

Proposed improvement:

Item: C12
Clause: 16.1. Access transparency
Importance:
Rationale:

Stubs are engineering viewpoint terms. Transparencies should be described using more general terms.

Proposed improvement:

Describe access transparency using functions, not engineering objects.

D. Federation, community, domain, interceptors

Item: D1
Clause: Part2/10.3, Part3/5.1.7, 5.1.8
Rationale:

Part 2 and Part 3 define terms "domain", "community", and "federation". However, there seems to be different views in Part 2 and Part 3 about the usage of these terms. This has caused problems for example in Trading function group.

The following interpretation has been used:

DOMAIN

Domain is an area in a technical sense. This area is restricted by a shared rule. In ODP−language it could be said that everything in a domain is of same type. Domain could be for example a security domain, where everyone is allowed or denied same service. There could also exist a communication domain, where everybody uses the same protocol (== "protocol domain").

Domain does not restrict any community issues.

COMMUNITY

Community is a group of members with a common activity. Everybody in a community work for a shared goal. Community members may be situated at different domains.

FEDERATION

When objects at different domains communicate, it is called federation. For example, two objects in a security domain may need an interceptor (protocol translator) because they are situated on different communication domains (they use different protocol).

Federation means crossing domain boundaries, one or many. Each domain boundary requires an interceptor (pair).

INTERWORKING

When objects at different communities communicate, it is called interworking.

Interworking means, that objects already playing some roles within their communities assume additional roles that bring them into an additional community.

Usually interworking or cooperation is referred to when an application is distributed. The objects in the distributed system play the same role towards the local clients, but when they have to synchronize or use other distribution dependent features, they assume additional roles.

Proposed improvement:

Change 5.1.8:

5.1.8. <X>−Federation:

<X>−federation is a community with objective X, such that it supports transparency for all <Y>−domain boundaries between members of <X>−community.

Item: D2
Clause: 8.1.11.
Importance: Editorial
Rationale:

An interceptor is needed to cross a domain boundary. Domains are introduced as <X>−domains.

Proposed improvement:

Change Interceptor to <X>−Interceptor.