Finland votes NO for the progression of ISO/IEC JTC1/SC7/WG17 W17N0080 Open Distributed Processing – Reference Model – Enterprise Language working document to a CD. We have the following comments.

FIN-1	lCat G	Page	Sect	Para	Line

The working document is not clear in the definition and the use of terms role, role type, community, community type, community template, and enterprise object. Our view is as follows:

1. A community specification discusses various roles as placeholders for potentially multiple objects. An object populating a role must be of object type that is compatible with the role type for that specific role. The relationship between role and object is not a type-instance relationship.
2. Role can be assigned with a cardinality. The cardinality treshold values express the limits within which the number of “object placeholders” of the given role type may vary within the community. Cardinalities could as well be expressed as indirect population criteria referencing to properties of the community instead of only properties of the objects.
3. A community specification can be expressed either as a community type or as a community template. In case of community type specification, the community (instance) where enterprise objects interact is established by populating the roles based on the criteria expressed as role type and population criteria. In case of community template, the population process may be implemented as an object instantiation process, where role specifications are used as object templates. However, we consider this as a special case of the first one leading to fixed communities.
4. Communities are allowed to modify themselves by deleting and adding roles of already specified types. The population process provides a mechanism for dynamic changes on the community participating enterprise objects. The obligations of the community are unchanged independent on whether a role is populated at a given time or unpopulated. Obligations can only be changed via explicit community changes.

Suggested text improvements founded on this rationale below.

FIN-2	Cat G	Page	Sect	Para	Line

The working document is not clear in the ways allowed for community establishment. The population process is not adequately described, instead frequent references to instantiation of enterprise objects and community instances can be found. As stated in FIN-1: 3, we consider the population process the essential one.

Furthermore, the specification method for communities appears to confuse between roles and objects, especially putting too much emphasis on identifying enterprise objects instead of roles. The alternativeways of specifying a community need to be explained separately.

Suggested text improvements below.

The working document is not clear in specifying the possible relationships between communities, nor in describing the possible relationships between roles. The following view to these relationships gives a rationale for text suggestions to follow.

The relationships between enterprise specifications can be divided into two essentially different categories:

A) relationships that deal with the behaviour of the specified communities; and

B) relationships that deal with the production of the enterprise specifications.

Both categories comprise several sub-cased where modifications to the enterprise properties (e.g., enterprise policies) are managed differently.

In addition,

C) communities may interfere with each other at the level of object instances populating roles.

A) The behaviour of two or more communities can be related via two basic mechanisms:

1. Communities can be designed to interact.

Two or more communities can be interlinked by designing how their policies, interactions, interfaces, and behaviours interleave.

The purpose of creating independent communities can be modelling of autonomous domains or modelling of independent service provision.

Examples:

- the legal domains involved in international goods markets;
- a company out-sources the provision of an information service.

Cooperation between communities is in this case captured at design time, and requires that the specifications of all involved communities are created, controlled, and modified by the same authority.

Any modification to the community specifications need to be taken to all involved specifications equally, to preserve consistency. This must either be done manually, or can be supported by tools. However, the result of the modifications is a new set of specifications, and a new set of communities.

A specific form of interaction can be created between communities by requiring that same object fulfils specific roles in the communities interacting through the shared object.

2. Communities can enter a cooperation state through a binding process at the system run-time.

Two or more communities can be interlinked through a match-making process that considers they policies, interacting capabilities, interfaces and behaviour descriptions.

The involved communities are in this case designed separately without a common controlling authority. Furthermore, the cooperation between the communities does not necessarily create a common authority for them.

This case is modelled with the help of community equivalent object (CEO): A community specification is considered as a property description of a class of object instances that are capable of fulfilling the joint responsibilities of all roles in the community.

A role in any community specification can thus validly be fulfilled by a CEO.

As the binding process can be late and dynamic, a single community specification can be created and modified independently from others. The consistency requirements are checked at the binding time.

Modifications to a single community (e.g., policy change) is propagated to other involved communities only if that is separately agreed as an activity between the various communities. Such an activity may be defined as a peer-to-peer negotiation or as a authority-to-subordinate relationship.

The example cases for dynamic inter-linkage are similar to the examples given in case 1.

B) In the production of enterprise specifications,

the interesting relationships include: nesting of communities, and reuse of role specifications. These cases are closely related to A.1.

Nesting of communities means, that a partial community is textually separated as a community of it's own, and the textual representation can be reused (by a specification tool) as part of multiple, potentially unrelated community specifications.

Reuse of role specifications works similarly, but in this case the isolated textual representation covers only a single role.

The role and community specifications can be private to a specification and development environment, or they can be stored to a repository that can be shared by a wider audience of several development environments and groups.

C) Object instances may populate roles in more than one community simultaneous.

In such a case, the behaviour of the object is restricted by the policies of each community separately. An object should not enter a community (bound to one, designed to be part of one) when contradictory requirements occur. However, the policies of the simultaneously participated communities are not necessarily consistent, but the object may end up into a contradictory situation where it necessarily causes a failure in one of the communities it participates.

This case is closely related to the role population process, that also plays an important role in A.2.

FIN-4 Cat G Page Sect Para Line

The concepts presented for processes and the relationships between processes, actions and roles are unclear. We consider that the following concepts are essential and should be used.

Behaviour (of an object): A collection of actions with a set of constraints on when they may occur.

Behaviour is a combined set of all possible sequences of actions (internal and interactions) that the object is cabable to participate, given the right environmental circumstances.

Composition of behaviours (Part 2, 9.1): A combination of two or more behaviours yielding a new behaviour. The characteristics of the resulting behaviour are determined by the behaviours being combined and the way they are combined (sequential composition, concurrent composition, interleaving, concealment of actions).

Cobehaviour is a composition of behaviours where the interactions of two or more objects are shown, and the internal actions hidden. The interactions visible cover both the interactions amongst those objects themselves and the interactions with the environment of the objects under composition.

Role: An identified set of activities relevant for the cobehaviour of a community.

Activity: A single-headed directed acyclic graph of actions, where occurrence of each action in the graph is made possible by the occurrence of all immediately preceding actions.

Single-headedness is a result of taking a single object in a given state as a starting point. Thus, the behaviour graph is restricted (partitioned) to separate subgraphs, each presenting a case where a certain environmental or internal prerequisite has been met.

Notes from Part 2: Action and activity are degenerate cases of behaviour. In general, several sequences of interactions are consistent with a given behaviour.

Action is anything that happens, according to Part 2.

Process: An identified subgraph of the community cobehaviour limited between interactions between the community and its environment (trigger and result).

Task: A labeled graph of actions. A task can be related either to cobehaviours (and thus, processes) or to activities (and thus, roles), and used for locating corresponding interactions presented in both role and process based approaches. The tasks need not to be identical, but some sort of equivalence class is more suitable.

Some of these are already present in Part 2 and 3 of ODP RM, and should not be repeated in this docume.nt The suggested definitions and text changes are given below.

FIN-5	Cat E	Page	Sect 5	Para 2, 4	Line

Replace by

“The enterprise language provides the vocabulary and constructs to specify the purpose, scope and policies for an ODP system in terms that are meaningful for the stakeholders for that system. An enterprise specification describes the behaviour of the system within the with which it interacts. Such an environment can be a technical environment (e.g., software and hardware environment of a service component) or a social or business organisation (e.g., a group of co-operating companies, particular service inside a company). “

and by

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

FIN-6 Cat E Page Sect 6.1.2 Para Line

Add note:

Note: A range of different kind of communities can be described. Some of them have a definite goal that can be reached and a termination condition can be captured as a objective statement. In a more general case, communities are intended to aspire an improved state without a termination condition, thus only preferences for particular outcomes of processes within the community can be stated.

FIN-7 Cat E Page Sect 6.2.4 Para Line

Keep 6.2.4.1 and the notes of 6.2.4.2; delete the rest of the text in 6.2.4.

FIN-8 Cat E Page Sect 6.2.3. Para Line

Delete. Use of normal English. Contracting party as a term does not really reveal that the commitment has already been made, it might as well be on the way still.

FIN-9 Cat TH Page Sect 6.3.1, 6.3.2. Para Line

Replace "Role (of a community)" by "Role (in a community)" in 6.3.2. The genetive form is ambiguous.

Replace definition in 6.3.2 by

> ***Role:*** An identified set of activities relevant for the cobehaviour of a community.

Replace the (empty) definition of behaviour in 6.3.1. by

> ***Cobehaviour***: A composition of behaviours where the interactions of two or more objects are shown, and the internal actions hidden. The interactions visible cover both the interactions amongst those objects themselves and the interactions with the environment of the objects under composition.

FIN-10 Cat TH Page Sect 6.3.3 Para Line

Replace the definition with the following and delete the three notes.

> **Process:** An identified subgraph of the community cobehaviour limited between interactions between the community and its environment (trigger and result).

FIN-11 Cat TH Page Sect 6.3.4, 6.3.5 Para Line

Replace the definition of task with the following.

> **Task:** A labeled graph of actions. A task can be related either to cobehaviours (and thus, processes) or to activities (and thus, roles), and used for locating corresponding interactions presented in both role and process based approaches. The tasks need not to be identical, but some sort of equivalence class is more suitable.

Delete the definition of step.

FIN-12 Cat E Page Sect 6.4.3, 6.4.4, 6.4.5.2.2 – 6.4.5.2.5

Delete.

FIN-13 Cat E Page Sect 6.5, 6.7 Para Line

Move 6.7.2 Commitment into policy concepts, making it 6.5.4.

Remove 6.7 Force concepts.

FIN-14 Cat TH Page Sect 7.1 Para Line

Replace by:

> In an enterprise specification, an ODP system and the environment in which it operates are represented as a community. The objectives and scope of the ODP system are defined in terms of the roles it fulfils within a community and policy statements about those roles.
>
> Communities can be defined either as community types or even as community templates.
>
> An enterprise specification may thus include the specifications of
>
> - A community and the direct environment it interacts with, and
> - Any other communities of which the system or its parts are members.
>
> A community specification defines the obligations, permissions, prohibitions and authorisation of each of the community members.

Notes 2 and 3 can be kept.

FIN-15 Cat E Page Sect 7.2 Para Line

Add opening sentences before 7.2.1:

> A community is a configuration of enterprise objects interacting with each other, governed by policies committed to that community. A community is formed by populating a community specification according to its population policy. A community involves a contract, an agreement between enterprise objects, about a shared objective and shared contstraints on interactions.

FIN-16 Cat E Page Sect 7.2.1 Para Line

Replace by

> A community specification is given in terms of
>
> - Roles to be fulfilled by enterprise objects,
> - Policies governing the activities related to each role,
> - Processes which take place in the community,
> - Policies governing the cobehaviour of objects related to each process,
> - Epochs describing changes in the presence of various roles within the community and changes in the configuration of those roles,
> - Population policies governing the assingment of enterprise objects to each role,
> - Policies relating to environment contracts governing objects in the community.
>
> In general, community type specifications form a type set in which various relationships between the types may be present. Corresponding cross references between the specifications concern may occur.
>
> Roles can be described by role types. Role types can have various relationships, e.g., subtype hierarchies.

FIN-17 Cat TH Page Sect 7.2.2 Para Line

Replace by

> A community specification can be populated by assigning an enterprise object to each role in the specification. The object type of the object that fills the role must be consistent with the role type. However, the role/object relationship is not a type/instance relationship. Furthermore, the enterprise object selected to play a certain role must fulfil the requirements of the population policy.
>
> > Note: As a special case, an instantiation process can be used for populating the community. In such a case, the role type is interpreted as an object template. However, the created communities are fixed both in respect of members, and of configuration.

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

An enterprise object may become a member of a community because

- The community specification provides that the community includes the object,
- The object is made part of the community at the time of community creation, or
- The object becomes a part of the community as a result of dynamic changes in the configuration of the community.

FIN-18 Cat E Page Sect 7.2.2, 7.8.2, 7.8.3 Para Line

Move sections 7.8.2 and 7.8.3 to form the final paragraphs in 7.2.2.

FIN-19 Cat TH Page Sect 7.2.3 Para Line

Rationale: See xxx. We doubt the verifiablity of the transitivity properties claimed in paragraph 3; furthermore, we are not sure that the property would be always beneficial.

Replace the section with the following text but keep the final note with examples.

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

Community specifications can be related in various ways, including relationships where

- Community specifications are interlinked by desinging how their policies, interactions, interfaces and behaviours interleave.

  Note: The purpose of creating independent communities can be modelling of autonomous domains or modelling of independent service provision.

  Note: In case of modelling federated communities, objects and roles involved may have contradicting requirements and constraints on their cobehaviour. If such a possibility exists, the specifications should include an explicit mechanism for resolving the contradicting situation.

- Community specifications can be reused and nested within specification tools.

Communities can be related in various ways, including relationships where

- A community (CEO, community equivalent object) fulfils a single role in another community. Two or more communities can be thus interlinked through a match-making process that considers their policies, interacting cababilities, interfaces and behaviour descriptions. The role to be fulfilled by thet CEO can be either a core role or an environment role.
- The same object fulfils roles in multiple communities. If communities require that the same object fulfills specific roles, the shared object can be used as a specific form of interaction between the communities.

> Note: Not all situations where an enterprise object fulfils roles in multiple communities are relevant. The situation may well be accidental and carry no intended obligations for the object. In contrary, such a case may contradict the security policy of some community.

Role specifications, role type specifications, process specifications, and policy specifications can be related in various ways, including relationships where

- specifications can be reused and nested within specification tools.

Communities with a need to interact are clearly parts of a larger, outer community, thus common policies will apply.

FIN-20 Cat E Page Sect 7.3 Para 1 Line

Replace by

Every community has one objective, which is defined in its contract, and can be refined into a collection of sub-objectives. The objective of the community is not necessarily a reachable goal but an aspiration towards some less specific aim.

FIN-21 Cat E Page Sect 7.3 Para 2 Line 3

Replace by

Policies describe the parts of the community objective which cannot be ultimately reached, but which influence decisions withing behaviour, on case-by-case basis, in order to steer the behaviour towards the objective.

FIN-22 Cat E Page Sect 7.4.1 Para 1 Line 1

Replace by

An object *denoted* by a community specification is involved …

FIN-23 Cat E Page Sect 7.4.2 Para Line

Replace by

Role-based approach: The cobehaviour of the community is partitioned into a set of object placeholders, roles, giving restrictions for the suitable enterprise objects to play the role, and obligations for the object within the community. For each role, only the required object behaviour is described, not the total cobehaviour of the community nor the behaviour of interacting partners.

Replace by

> Process-based approach: The cobehaviour of the community is partitioned into a set of processes, each describing how the community reacts to a trigger and reaches a result, and in this way achieves some particular sub-objective in the community. For each process, the required interactions between core objects and environment objects are described, not hiding the internal communication of the community.

Delete para 3-6 (Role behaviour decomposes …. not be a step.)

Replace step by task.

FIN-24 Cat E Page Sect 7.4.3, 7.4.4 Para 3 Line 1

Replace by

> When a community is established, one or more objects is associated with …

Add

> Roles can have cardinalities assingned to them. This is a shorthand for expressing that the same type of role can be multiply populated simultaneously.

Delete 7.4.3.1, 7.4.3.2 and 7.4.3.3 , 7.4.4

FIN-25 Cat E Page Sect Annex B and parking lot Para Line

Delete.

FIN-26 Cat TH Page Sect 10.3 Para Line

The correspondence rules between enterprise language and other ODP viewpoint languages are not defined in terms of ODP vocabulary. In addition, the text related to corresponding concepts has been degenerated too far by iterated corrections.

- In the illustrations, for enterprise viewpoint, the following boxes should be used:
  - community
  - Objective
  - Scoping statement
  - Role
  - Process
  - Task

  - Actor role
  - Artefact role
  - Policy
- For information viewpoint, the following concepts/boxes should be used:
  - Information object
  - Dynamic schema
  - Static schema
  - Invariant schema
- The correspondences between enterprise viewpoint concepts and information viewpoint concepts include
  - One or more information objects can represent the information content of an enterprise object which fulfils any kind of role in the enterprise specification.
  - Invariant, static and dynamic schemata of information objects are governed by the enterprise viewpoint policies. The policies each information object must obey are those related to the whole community, those related to the enterprise object involved, and activities of that enterprise object as far as the information object is participating the activity.
  - A dynamic schema is associated with a task.
- For computational viewpoint, the following concepts/boxes should be used:
  - Computational object behaviour
  - Computational interaction
  - Computational interface
  - Binding object
- The correspondences between enterprise viewpoint concepts and computational viewpoint concepts include
  - A computational object may include multiple computational interfaces each of which represent a computational object behaviour. One or more computational interfaces can represent the computations related to an enterprise object.
  - One or more interactions of a computational object are related to a task.
  - Computational object behaviour is restricted by policies.
  - Binding object is related to cobehaviour of enterprise objects.
- For engineering viewpoint, the following concepts/boxes should be used:
  - Basic engineering object
  - Node
  - Interceptor

- The correspondences between enterprise viewpoint concepts and engineering viewpoint concepts include
  - One or more basic engineering objects may represent an enterprise object.
  - An enterprise object may locate at one or more nodes.
  - A domain is a specific community type in the enterprise language. One or more interceptors may be needed to implement a domain boundary.

FIN-27 Cat TH Page Sect Annex A Para Line

The BNF representation of the enterprise specification structure does not conform with the text.

Furthermore, the population constraints have been moved into a too detailed level (actions) to be useful. Population constraints are not any more clear from the text either, and should be added.

Move <constraint>*<involves statement>* into <role-object assignment>.

Figure A-1 should be updated:

- Objective and process should have N:1
- Step and action should be removed
- SuperWho should be removed or clarified, RoleFiller as well
- Behaviour and SuperWho should have no relationship
- Cobehaviour should be defined
- Behaviour should have a relationship with task

FIN-28 Cat E Page Sect 7.5, 8 Para Line

Replace section 7.5, before 7.5.1, by section 8, thus removing 8.

Keep the first paragraph of 7.5.1, but remove the rest of the section.

FIN-29 Cat E Page Sect 7.5.2 Para Line

Replace the subtitle Ownership by “Policy makers” and continue 7.5 directly. Replace the text by

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

FIN-30 Cat E Page Sect 7.5.3 Para Line

Add "authorization" to the title. Add subsection Authorization

If an enterprise object is authorised to an action, the target of that action has no permission to deny the action to take place. If denial occurs, that is a failure and a violation against the community contract.

FIN-31 Cat E Page Sect 7.5.3.1 Para 3 Line 24-34

Delete the last paragraph about deontic logic, and the temporary note.

FIN-32 Cat E Page Sect 7.5.3.3 Para Line

Replace "Standing obligations … violation condition" by

Standing obligations are always obligations where enabling conditions are always true, thus any interaction must conform to the obligation in addition to other constraints on it.

FIN-33 Cat TL Page Sect 7.5.6.1 Para Line

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

- Associated with a particular target role in this interaction, or
- Associated with the interaction as a whole.

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

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

FIN-34 Cat E Page Sect 7.5.6.2 Para Line

The purpose of this section is unclear. Unless the intention can be clarified, the section should be deleted.

FIN-35 Cat E Page Sect 7.7 Para Line

See comment xxx. The proposed text for 7.2. captures the ideas of current 7.7. Therefore 7.7 can be deleted.

FIN-36 Cat E Page Sect 7.9, 7.10 Para Line

Remove.

FIN-37 Cat E Page Sect 7.5.4 Para Line

Delete. The idea has already been inserted into the text by comments xxx, yyy.