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

ISO/IEC JTC1/SC7 /N3257

2005-06-11

Document Type	Study Group Report
Title	Report of the Study group on the Revision of RM-ODP
Source	Study Group
Project	
Status	Final
Reference	
Action ID	FYI or ACT
Due Date	
Distribution	AG
No. of Pages	28
Note	

Address reply to: ISO/IEC JTC1/SC7 Secretariat
École de technologie supérieure – Département de génie électrique
1100 Notre Dame Ouest, Montréal, Québec Canada H3C 1K3
secretariat@jtc1-sc7.org

www.jtc1-sc7.org

ISO/IEC JTC1/SC7/WG19

Title: Report of SC7 Study Group on the Revision of RM-ODP

Source: SC7 Study Group on the Revision of RM-ODP, Helsinki, May 2005

Status: For review and comment by National Bodies.

Date: May 26, 2005

1. Background

1.1 SC07 Plenary Resolutions

The Study Group was established by Resolution 731 of the Montréal Plenary of SC7:

> *JTC1/SC7 instructs its Secretariat to establish a Study Group to gather requirements for the revision of ITU-T Rec. X.901-4|ISO/IEC 10746 Reference Model of Open Distributed Processing (RM-ODP).*
>
> *The detailed terms of reference of this study group will be circulated to SC7 before the end of July 2003.*
>
> *The study group will be led by Mr. Bryan Wood, UK. The membership of the Study Group will consist initially of*

Mr. Jean Bérubé	*Canada/SC32*
Mr. Tom Rutt	*US*
Mr. Akira Tanaka	*Japan*
Mr. Jonathan Billington	*Australia*
Mr. Antonio Vallecillo	*Spain*
Mr. Arve Meisingset	*Norway/ITU-T*
Mr. Sandy Tyndale-Biscoe	*U.K.*

> *Further nominations must be sent to the JTC 1/SC7 Secretariat by 2003-09-15.*
>
> *The study will last two years. An interim report will be given at the Brisbane 2004 plenary and a recommendation to initiate a project or not on that topic at the following plenary.*
>
> *The Study Group will liaise with ITU-T SG17 and OMG. It is understood that the study group will need to be re-created at the Brisbane Plenary.*

1.2 Open Distributed Processing and RM-ODP

The objective of ODP standardization is the development of standards that allow the benefits of distributing information processing services to be realized in an environment of heterogeneous IT resources and multiple organizational domains.

These standards address constraints on system specification and the provision of a system infrastructure that accommodate difficulties inherent in the design and programming of distributed systems.

Distributed systems are important because there is a growing need to interconnect information processing systems. This need arises because of organizational trends such as downsizing, which demand the exchange of information both between groups within an organization and between cooperating organizations. Advances in technology are making it

possible to respond to these trends by giving increasing importance to information service networks and personal workstations, and by permitting the construction of applications distributed across large configurations of interconnected systems.

ODP standardization has four fundamental elements:

- an object modelling approach to system specification;
- the specification of a system in terms of separate but interrelated viewpoint specifications;
- the definition of a system infrastructure providing distribution transparencies for system applications;
- a framework for assessing system conformance.

2. Scope & Objectives of the Study Group

2.1 Scope

The field of application of the study group is ODP standardization, which is the development of standards that allow the benefits of distributing information processing services to be realized in an environment of heterogeneous IT resources and multiple organizational domains.

Within that field of application, the study group is concerned with requirements for revision of the Reference Model of ODP, which provides a framework for ODP Standardization.

ITU-T Rec. X.901-4|ISO/IEC 10746 is the mature result of a large amount of technical effort. The Study Group is not expected to propose major restructuring or change, but to correct errors and take account of recent developments in the industry, so as to maintain its broad scope. However, if it becomes apparent during the study that there is merit in considering the standardization of alternative frameworks based on different structuring principles, the Study Group may recommend the establishment of new projects to that effect.

In proposing any update to the Reference Model, the group will take into account the need to remain consistent with existing standards based on it, such as the Enterprise Language, the ODP Trader, the Naming Framework and the standard for Interface References and Binding.

2.2 Objectives

The objective of the study group was the elicitation and analysis of requirements for the revision of the ODP Reference model, and the preparation of a set of recommendations to SC7 about what actions, if any are required.

3. Identification of Issues

Issues were identified and developed through a Workshop on ODP for Enterprise Computing at EDOC 2004 and inputs from NBs to the WG19 meetings in Malaga in November 2004 and Helsinki in May 2005. The results are given in the Annex to this report.

In the Annex, priorities are suggested for the issues in the range 1 to 3, where 1 is the highest. and effort is suggested on a scale of 1-10, were 10 is the highest.

4. Recommendations

The Study Group recommends that

1. That revision of ITU-T Recommendation X.901-904 | ISO/IEC IS 10746, Information Technology - Open Distributed Processing – Reference Model should be considered in order to address the issues that have been identified by the Study Group.

2. The revision should be planned in two stages, taking into account the priorities given to the requirements for revision and the estimated resource requirements.

Stage 1 would address:

Issue 1	Cat TE	Part 3 – 5.1.1, page 7, Community
Issue 2	Cat TE	Part 3 – 8.2.1, Figure 3, page 21, Channel rules
Issue 3	Cat TE	Part 3 Improved alignment of RM-ODP Part 3 with Part 2
Issue 4	Cat TR	Part 2 – 9.14, page 7, Role
Issue 5	Cat TR	Part 2 – 8.3, page 4, Action
Issue 6	Cat TR	Part 2 – 11.2, pages 10-11, Policy concepts
Issue 8	Cat TR	Part 2, Components
Issue 9	Cat TR	Organisational units
Issue 10	Cat TR	Part 2 – 8.3 page 4, Additional definitions
Issue 11	Cat TR	Service concepts
Issue 13	Cat TR	Relationship between specification and instantiation
Issue 14	Cat TR	Part 2 – 15.3.2, Human/system interactions
Issue 15	Cat TR	Asynchronous, message-based interactions
Issue 16	Cat TR	Part 3 – 7.1.11, 7.1.12, 7.1.13 and other places, Number of parameter elements
Issue 17	Cat TR	Causalities
Issue 18	Cat	TR Signatures for action templates
Issue 19	CL	Part 3 – 7.1.5 and 7.2.2.5, Flow and use of signals to build interactions
Issue 20	Cat CL	Part 3 computational interaction types (interactions without causalities, different patterns, etc.)

Issue 21	Cat CL	Explicit architectures for binding objects
Issue 24	Cat CL	There is a need to clarify the nature of the relationship between computational and engineering viewpoints
Issue 25	Cat TR	There is a need to clarify the nature of the technology viewpoint and its difference from the engineering viewpoint
Issue 26	Cat TR	Infrastructure and abstract platforms

Stage 2 would address:

Issue 7	Cat CL Part 2,	Encapsulation and refinement
Issue 12	Cat TR Part 2,	Definitions relating to evolution in time
Issue 22	Cat CL	Composition/decomposition of objects and interfaces
Issue 23	Cat TR	There is a need to clarify the nature of correspondences that apply between viewpoints
Issue 27	Cat TR	Legacy systems
Issue 28	Cat CL/TR	ODP Guide – RM-ODP Part 1

3. An NWI proposal should be developed immediately for the Stage 1 revision.

Annex

Issues for consideration for a revision of the RM-ODP

The sources of the issues are:

MAL-010 Japan issues for consideration for review of the RM-ODP

MAL-013 UK issues for consideration for review of the RM-ODP

MAL-033 Report of Workshop on ODP for Enterprise Computing (WODPEC 2004)

HEL-009 UK Comment on MAL-034

This document uses the following categorisation of issues.

Category	Description	Impact
G	General	Applicable generally i.e. in multiple places throughout
E	Editorial	A cosmetic change, including typographical and grammatical
CL	Clarification	A change to clarify the intent of the text
TE	Technical Error	A change to correct an error in the text
TR	Technical Refinement	A change to extend or refine concepts

Priorities are suggested for the issues in the range 1 to 3, where 1 is the highest.

[13], [14], [15] and [16] are referred to collectively as the RM-ODP. Clauses in individual Parts of the RM-ODP are referred to in the form "Part n – m.m" – a reference to clause m.m of RM-ODP Part n.

Issue 1 Cat TE Part 3 – 5.1.1, page 7, Community

Priority: 1 – Effort: 1

Rationale

The Enterprise language standard, ISO/IEC 15414, has used this definition but found it necessary to change the second sentence from "The objective is expressed as a contract which expresses how the objective can be met." to "The objective is expressed in a contract which expresses how the objective can be met."

Proposal

Change second sentence of Part 3 – 5.1.1 to:

> The objective is expressed in a contract, which expresses how the objective can be met.

Issue 2 Cat TE Part 3 – 8.2.1, Figure 3, page 21, Channel rules

Priority: 1 – Effort: 1

Rationale

Edward A. Feustel of the Institute for Defense Analyses has pointed out that in figure 3 the Interceptor P2 to P3 connects a Protocol 2 object to a Protocol 2 object whereas it should connect a Protocol 2 object to a Protocol 3 object.

Proposal

In Part 3 – Clause 8.2.1, Figure 3, replace the box connected to box labelled "Interceptor P2 to P3" and labelled "Protocol 2 object" by a box labelled "Protocol 3 object".

Issue 3 Cat TE Part 3 Improved alignment of RM-ODP Part 3 with Part 2

Priority 1 – Effort: 4

Rationale

Although the concept definitions in Part 2 are quite stable, they are not always used to best advantage in related standards. In particular, because Part 3 (Architecture) was developed in parallel with Part 2 (Foundations), there are places where usage in Part 3 is incorrect or where rules could be expressed more precisely by making best use of the Part 2 concepts.

Proposal

Review and correct the alignment of Part 3 with Part 2.

Issue 4 Cat TR Part 2 – 9.14, page 7, Role

Priority: 1 – Effort: 3

Rationale

a) There is a problem with the definition of *role*.

Part 2 defines a *role* as "an identifier for a behaviour, which may appear as a parameter in a template for a composite object, and which is associated with one of the component objects of the composite object". It has become clear that this does not, in its present form, convey the intent, which is indicated by the reference to *templates* and to the actualisation of parameters in the second paragraph of the definition.

The metaphor on which the *role* concept is based is theatrical. The text of a play is expressed in terms of lines and actions associated with various *roles*, which are declared initially in a cast-list. Putting the play on involves assigning actors to the various *roles*, although one actor

may play several minor *roles*, and the actor playing a *role* may change during the run of the production. Identifying the *roles* rather than the actors obviously makes the script more reusable.

The key idea is that some constraints on *system behaviour* are associated with *objects* dynamically as a consequence of an earlier part of the *behaviour*, such as performance of a piece of negotiation. However, although the potential *behaviour* can be referenced (and hence the talk of an identifier in the definition) it is not associable with an actual *object* until the *template* is instantiated and the *role* bound to a specific *object*.

b) There is a problem with usage rather than definition.

There has been widespread discussion in ODP circles of community-roles in the *enterprise language,* but unfortunately this has been expressed the term using *role* without qualification, leading to an implicit assumption that saying *role* implies community-role. As shown in (a), *role* is defined as a parameterization mechanism for *templates*, and so can potentially be applied to anything for which a *template* can be defined. Indeed, there are other places where the *role* concept is not just useful but is vital to making necessary distinctions in the *template* definition.

Perhaps the clearest present need is in the definition of *action templates*, particularly *interaction templates*. In an *interaction* between, say, a client and a server *object*, it is essential to know which is which. We can do this by saying that the two *objects* in this example fill client and server *roles* in the *interaction*, and by associating necessary properties and constraints with these *roles*. At one point in a *system's behaviour*, an *object* A can fill the buyer *role* in a purchase *interaction*, while *object* B fills the seller *role*, and later the *roles* can be reversed, so that B is the buyer and A is the seller. The richer the *interaction*, the more useful this approach is likely to be; it is particularly effective, for example, for expressing the capabilities and obligations associated with secure multi-way *interactions*, where capabilities or access permissions are clearly associated with a specific *role*. Another example is for distinguishing between actor and artifact *roles* in enterprise *interactions*.

See [2]

Proposal

a) The solution to the problem of misunderstandings lies in removing the idea of there being a parameter identifier in the *template's* behaviour to the explanatory note, and to focus the first paragraph of the definition on the idea of parameter substitution. Perhaps more importantly, there is a need to clarify the way the potential *behaviour* of an *object* is restricted when it is bound to a *role*, and this needs a proper framework for the discussion of potential *behaviour*.

b) The solution to the problem of usage is for users of the *role* concept to be encouraged always to qualify their use of *role* with the *template type* name, as in *action-role* and *community-role*.

Issue 5 Cat TR Part 2 – 8.3, page 4, Action

Priority: 1 – Effort: 4

Rationale

1. The significance of the use of the term *action* in a specification derives from to the way in which the RM-ODP talks about the real world. The RM-ODP "concentrates on descriptions

that can be tested by experiment – by observation. This places emphasis on describing behaviour made up of *actions* which model things that happen in the real world. Objects, modelling real world entities, emerge in this framework as a descriptive tool for organising the expression of behaviour." (see Specification and Implementation in ODP, Peter F Linington and William F. Frank, WOODPECKER 2001, http://www.cs.kent.ac.uk/people/staff/pfl/papers/pfl-woodpecker-01.pdf). The way in which *action*, *object* etc are defined in Part 2 is consistent with this view and the expression of these concepts in an ODP model should be interpretable in a way that is consistent with this view.

2. Clarification of the relation of the term *action* to UML Action Semantics is an issue for work on the use of UML for ODP system specifications (ISO/IEC 19793).

See [7].

Proposal

Provide clarification in Part 2 of the modelling view taken by the RM-ODP, along the lines of the Rationale.

Issue 6 Cat TR Part 2 – 11.2, pages 10-11, Policy concepts

Priority 2 – Effort: 3

Rationale

a) The current definition of a *policy* in Part 2 is very weak. It is just defined as "a set of rules related to a particular purpose", with an indication that the rules are expected to be expressed in deontic terms. There has been a great deal of work on the use of *policies*, particularly in various styles of policy-based management, since the creation of the RM-ODP, and the requirements are now much better understood.

b) Part 2 defines a number of deontic concepts, particularly *obligations*, *permissions* and *prohibitions*, but this part of the framework was produced before there had been much experience with their application in ODP. The result is that the definitions were taken directly from the Standard Deontic Logic (SDL), including a simple set of relations between the concepts, such as the assertion that a *permission* for something is an indication that there is not an *obligation* not to do it.

c) Policy concepts are used in Part 3 – 5 and in ISO/IEC 15414, Enterprise language. There is a need to address the possible applicability of policy concepts in other viewpoint specifications.

See [2], [5] and [8]

Proposal

a) Two requirements should be specified as necessary for a rule to be considered a *policy*.

i. There must be some element of choice associated with any *policy*. *Policies* are identified in a specification wherever it is recognised that a rule may need to be changed during the lifetime of the specification; selecting a structure for the specification that emphasises the scope of the *policy* makes it easier to modify it without wholesale revision, and allows the likelihood of change to be reflected in the implementation. This is generally done by encapsulating associated decisions as the behaviour of a distinct *computational* or *engineering object* that can be replaced

whenever the *policy* is changed. Thus a rule that is universally true, and cannot be changed without wholesale replacement of the specification, is not a *policy*. The speed of light is not a *policy*, but the setting of interest on credit at a certain percentage above base rate is. Whether an organization operates with the status of a charity either might or might not be a *policy*, depending on whether the specifiers foresaw the possibility that the status might change and planned for it.

ii. The specifiers who identify it should generally wish to limit the range of *behaviour* that would be acceptable; this gives rise to the idea of a policy envelope [11], which limits the range of *behaviour* any particular *policy* is allowed to specify. Knowledge of the *policy* envelope allows the verification of invariants on the specification that are independent of the particular *policy* in use at any particular instant.

b) While there is nothing wrong with the definitions from the SDL, experience has shown that the SDL approach is somewhat brittle for enterprise modelling, and it would be better to take a less prescriptive approach in Part 2, allowing, for example, a style of modelling based on Utilitarianism to be exploited if it proves effective [12].

c) Provide text on the applicability of policy concepts in viewpoint specifications other than the enterprise specifications.

d) Work on ISO/IEC 19793, Use of UML for ODP system specifications, has raised issues which need to evaluated in the context in this issue.

Issue 7 Cat CL Part 2, Encapsulation and refinement

Priority 2 – Effort: 7

Rationale

Part 2 explains *encapsulation* as the property of an *object* such that it can only have its *state* changed by *interactions* or internal *actions* expressed in the model. As such, the definition is more related to completeness of description than to level of abstraction. Indeed, the inability to guarantee encapsulation on structural refinement is one of the problems in security analysis, for example, since structural refinements may introduce backdoors, and it is difficult to apply constraints to the refinement process that prevent this.

See [2].

Proposal

The concepts of *encapsulation* and *refinement* require clarification.

Issue 8 Cat TR Part 2, Components

Priority 1 – Effort: 3

Rationale

One of the significant changes in the last ten years has been the growth of interest in component-oriented architectures, where, taking for example the CORBA Components 3.0 Specification, the key properties of a component are:

a) encapsulation;

b) interactions at ports; the ports can be specialised as facets, receptacles, event sources, event sinks or attributes;

c) a component equivalent interface that provides metadata, navigation and control for the component;

d) an associated component home interface, representing a container in which components of the given type can be instantiated.

Thus, there is a need to determine whether the Part 2 should include a general definition of what a component is and how it can modelled.

See [2] and Issue 9

Proposal

All of the properties of a component can be modelled using the existing Part 2 concepts. The general concept of *interaction* is rich enough to be refined into any of the defined port types, and the equivalent and home *interfaces* are just conventional *computational interfaces*. The container property requires a specialisation of the *object* concept to make explicit the relation to the instantiation of templates involved when it acts as a factory. Thus there is no need for extension of the Part 2 to incorporate a computational or engineering component model into the architecture. However, there could be merit in adding derived definitions for component and factory, making it clear how the existing definitions can be used to model them. Such derived definitions for component modelling may be added to Part 2 – 9 and may be refined in Part 3 – clause 7 (Computational viewpoint).

Issue 9 Cat TR Enterprise Services

Priority 2 – Effort: 3

Rationale

Recent development in business-to-business systems has accelerated the standardization of message exchange protocols and the contents. If we look at enterprise systems participating in this business-to-business interaction, each participant could be seen as a large something (called, for convenience, “enterprise service”) exchanging information or data. One of the differences between this business-to-business interaction and object-based interaction is how behaviour is built into the messages exchanged. In business-to-business interaction, since both of the participating enterprises may use different technology for their systems, it is not appropriate to include, for example, object-reference or externalised object data in the message, because the systems have no shared knowledge other than information exchanged (they are loosely coupled). The message just contains information, and the format is usually defined in, for example, XML schema or something similar. To better represent this interaction model, there is a need to introduce a new concept such as “enterprise service”.

If we consider how this enterprise service is constructed, we may need to look at the internal structure of the system, which may be a composition of various enterprise services working together. It would be a welcome for users to have a composite interface (for a specific interaction) provided by this enterprise service, which is an aggregation of the interfaces of enterprise services composing the enterprise service.

See [8]and Issue 8.

Proposal

- A ”enterprise service” concept should be introduced.
- Additional interfaces for asynchronous communication should be elaborated.
- Composition, currently defined only for object and behaviour, should be extended to cover “composite interface.”

Issue 10 Cat TR Part 2 – 8.3 page 4, Additional definitions

Priority 1 – Effort: 2

Rationale

There are a number of areas in which the current Part 2 omits material on the grounds that it is self evident or sufficiently obvious to be taken as read. Experience has shown that it is worth making even apparently well-understood concepts explicit if they are to be used in a formal way.

One example is the omission of terms in common technical usage or used as normal English, such as relationship or association, definitions of which should be included on the basis of significant usage in ODP specification, even if they seem obvious. Part 2 should include generic definitions consistent with, but less detailed and restrictive than those in the ISO General Relationship Model [18]. Similarly, recent development in Policy and Rules (e.g. business rules) may require updates to the standard, and also recent industry trends on software/enterprise architecture (patterns) may require different kind of updates to the standard.

Another example is the omission of ODP specific detail or logical consequences. Here one might consider the explicit definition of the concept of a *viewpoint correspondence*, which is not discussed when *viewpoint* is defined. It should, indeed, be obvious to everyone that a system is only defined if both the *viewpoints* and the correspondences between them are detailed to a sufficient level to unify the overall specification, but sets of *viewpoint* specifications are often published without clear statement of correspondences, and a more balanced set of definitions would help to get this message across.

See [2].

Proposal

Add:

a) non-exclusive definitions for common refinements of *interaction*, such as invocation, message transfer and event notification;

b) definitions for terms in common technical usage, such as relationship or association (to Part 2 – 8, Basic modelling concepts), rule (to Part 2 – 7, Basic linguistic concepts) and pattern or style (to Part 2 – 10, Organizational Concepts);

c) explicit definition of the concept of a *viewpoint correspondence*.

Issue 11 Cat TR Service concepts

Priority 1 – Effort: 2-4

Rationale

The term *service* is not defined in the RM-ODP. It occurs in Part 2 – 13.3.6 in a Note, and becomes defined through various other term definitions in the RM-ODP and the Trading Function standard [20]. However, as a summary, we can say that a *service* is an abstract processing step that either creates, modifies, or consumes information from the point of view of its environment. *Services* are made available at *interfaces* (seen from the *computational* or *engineering viewpoints*) and defined by the structural, behavioural and semantic rules of the *interaction* involved (seen from the *enterprise*, *information* and *computational viewpoints*).

In the specific context of B2B operations, *services* are provided by administrative *domains*, for example by enterprises, departments or any independent ICT systems. The *services* can be published by exporting *service offers* to a trading service. The *service offer* represents details of the behaviour of the service (what kind of application protocol needs to be followed using it), and other information further describing the nature of the service depending on the application domain. The structure is more detailed than can be found from the ODP trading function standard that requires *interface type name*, *interface reference*, and some attributes as name-value pairs. What is to be noted here is that the *service offer* captures aspects from all five viewpoints, either as descriptions of the *service* to be provided or as a requirement to be fulfilled by the environment in the subsequent contract. This structure differs from OWL-S and UDDI based solutions (e.g. [28]) by not addressing the grounding or the actual location of services. These locating aspects are only captured by the federation contract formed according the offers; the grounding aspects are considered private for each enterprise. Only the interoperability-related features are required.

See [8].

Proposal

Add terms and definitions for the concepts of service and service offerings covering:

- Service;
- relation of behaviour to service;
- specification of binding types/categories in service offers;
- non-functional aspects of service offers and mechanisms to handle them.

Issue 12 Cat TR Part 2, Definitions relating to evolution in time

Priority 3 – Effort: 7-8

Rationale

It has always been a principle in the development of ODP that the RM-ODP is neutral with respect to the methodologies to be applied, and maintaining this position gives the most broadly applicable framework. Nevertheless, it is clear that most systems will evolve over time, and the reference model needs to take this into account.

The dynamic evolution of models means that model elements may be added or changed or deleted at some point to evolve into the model in the next phase. This may be viewed as a part of the lifecycle of models or versioning of models. For example, in the *enterprise viewpoint language*, dynamic creation and deletion of communities are described. Other possibility includes dynamic addition of *enterprise objects*, *roles*, *processes*, and so on. If an

enterprise viewpoint model evolves in such a way, corresponding viewpoint models should also evolve. In the *information viewpoint*, *information objects* might be added or deleted, and defined schemata may also evolve. In the *computational viewpoint*, the whole model might change. It will certainly impact the *engineering* and *technology viewpoint* models. There is a need to discuss:

1) how these evolutions or changes (transition) can be described in each viewpoint, and

2) what concepts in which viewpoint are required to achieve this.

If successful, the result may apply to "as-is model" and "to-be model" in Enterprise Architecture today.

See [2], [5] and [8].

Proposal

The Part 2 includes the concept of an epoch to describe the way in which objects or configurations of objects evolve through a series of stages. This concept can also be used to describe the evolution of the specification itself. This allows a new version of a specification to describe, for example, how it might be introduced as a staged transition from the previous version. Nevertheless, there is a need for more explicit definitions relating to evolution in time, both for system behaviour (for example the lifecycle of *interfaces*) and, more generally, to enable for a set of ODP specifications to reflect changes of requirement and policy – possibly by the introduction of frame-based models such as one based on Kripke Frames (see [21] for an accessible review of these and other related systems able to support modal logics) as a basis for unified semantics.

Issue 13 Cat TR Relationship between specification and instantiation

Priority 2 – Effort: 4

Rationale

There is a need to enhance the ODP *conformance* model slightly so that it usable to distinguish between classes of use to which the specification is being put, in particular, there is a need to clarify the relationship between specification and instantiation

See [2] and [10].

Proposal

The current ODP *conformance* model describes the relations between the system specifier, the implementor and the tester, and describes how conformance is deduced from observation during testing, confirming that the implementation is consistent with the original specification. However, there is a need to introduce the role of system owner, so that statements of rights to implement and use the design embodied in the specification can be made and then interpreted during the testing process to guide the interpretations made by the tester.

Issue 14 Cat TR Part 2 – 15.3.2, Human/system interactions

Priority 2 – Effort: 2-3

Rationale

In the *viewpoint languages* ([15]and [17]) today, there seems to be no place for describing human-system *interaction*. Although Part 2 defines "perceptual reference point (see Part 2 15.3.2)," no standard has been developed focusing on the subject of human-machine interactions in the family of RM-ODP standards.

When we consider a human-machine *interaction*, it is important to provide users with transparencies. *Transparencies*, such as access, migration, location, replication transparencies, are defined in RM-ODP. These are also useful for considering user level *transparency*. Combinations of these *transparencies* may provide a new *transparency* to users by hiding e.g. *system* configuration or deployment. Another *transparency* for users is a transparency that hides the means of *interaction* with a *system*, such as equipment for interaction (e.g. web form/fax/email and voice operation), and places where *interaction* occurs (e.g. internationalization with the use of locale at an *interface*).

There are several factors that cannot be concealed, and thus are essential, for actualizing *transparency* for users at every computation stage. The candidate factors are a viewer, using languages, user's operations of computations and the purpose of using a *system*. The reason why a viewer is necessary is that a user as a human being cannot see directly inside computing data. Therefore, a way of conversion may be needed to explicitly define a viewer concept, although the construction of such a viewer is arbitrary. A user will need to use a language to execute *interactions*, thus support of such a language representation is necessary. User's operations on a *system* may be considered as a disturbance from an outer world to the *system*. Thus, *interactions* of a user cannot be concealed. As users intend to do something in the *environment* of an ODP *system*, the purpose is definitely perceived. The representation of purpose can be shown by using semantic related notions.

Note that there is also an issue regarding "interchange reference point," which may cover access methods/formats to external storage system.

These concerns can be addressed using existing concepts in Part 2 – 15.3.2 and 13 and it is not necessary to introduce new concepts into the RM-ODP to do so. However, there would be value in providing clarification of the handling of human/system interactions in computational and engineering specifications.

See [5].

Proposal

Add additional text to Part 3 to clarify how existing concepts can be used to address human/system interactions in computational and engineering specifications.

Issue 15 Cat TR Asynchronous, message-based interactions

Priority 2 – Effort: 2-3

Rationale

Asynchronous and message-based interactions can be modelled in the computational language:

1. Part 2 does not make any distinction between “interfaces for objects that provide services” and “interfaces for objects requiring services” i.e. it allows for either. Part 3, Clause 7 assumes both the provision and the requiring of services e.g. “7.1.12 Operation interface signature: An interface signature for an operation interface. An operation interface signature comprises a set of announcement and interrogation signatures as appropriate, one for each operation type in the interface, together with an indication of causality (client or server, but not both) for the interface as a whole, with respect to the object which instantiates the template.”

2. The concept of asynchronous operations is supported by announcements (Part 3: 7.1.3 **Announcement**: An interaction -- the invocation -- initiated by a client object resulting in the conveyance of information from that client object to a server object, requesting a function to be performed by that server object.)

3. Message-based interaction can be explicitly modelled by means of binding objects that can be specified to provide message queuing functionality.

However, these modes of operation, which are particularly significant in business-to-business scenarios, interaction are not explicitly discussed in Part 3: 7.

See [5].

Proposal

Text should be added to Part 2 and Part 3 to clarify these issues.

Issue 16 Cat TR Part 3 – 7.1.11, 7.1.12, 7.1.13 and other places, Number of parameter elements

Priority 2 – Effort: 2

Rationale

The purpose of the parameter element giving the number of parameter elements in the *interface signature* needs clarification, since the names and types of parameters must also be provided

The idea of signature is that it brings together all the syntactic information about an interaction seen in isolation (i.e. not including the context in which the interaction happens - what went before, and so on) It does not include anything about the semantics, in terms of what was done to give rise to the interaction, or as a result of receiving it. The text in Part 3, 7.1.11 is an attempt to illustrate what "all the syntactic information" might mean by giving examples. In most cases, the list given is redundant, since if the actual parameters are listed they can be counted, however, it is necessary to look at the parameters to distinguish different flavours of operation with the same name (e.g. the artist object and the gunfighter object, both of which support the operation "draw"). However, it is difficult to cover all the complexity in

a brief list, particularly the distinction between the signature of an operation that has happened in a trace and an operation that is expected to happen in the future, because possible operations may involve the concept of optional parameters, so the situation gets more complicated.

Thinking in terms of templates, perhaps with IDL as an example in mind, then the number of parameters in a particular instance may matter, although in that case different allowed combinations of parameters would lead to alternatives in the specification.

Proposal

No change should be made to the normative text but some explanatory text should be added in a Note. This Note should clarify whether or not it is mandatory to include the number of parameter elements.

Issue 17 Cat TR Causalities

Priority 1 – Effort: 2

Rationale

Part 2 – 13.3 states that "the identification of *causality* allows the categorization of *roles* of interacting *objects*". Furthermore, that clause provides "a basic set of *roles*" and specifies that a "*causality* implies a constraint on each *behaviour* of the participating *objects* while they are interacting".

At the same time, Part 3 – 7.1 defines where the indication of the *causality* must be defined in each case, and for each element. For *signal interfaces*, their *interface signatures* "comprise a set of finite *action templates*, one for each type of *signal* in the *interface*. Each *action template* comprises the name of that *signal,* the number, name and types of its parameters and an indication of *causality* with respect to the *object* which instantiates the *template*". It is the same for *stream interfaces*. However, for *operation interfaces*, *causalities* are not treated in the same way. For *operation interfaces*, the *operation interface signature* comprises, apart from a set of *announcement* and *interrogation signatures*, as appropriate, the indication of *causality* for the *interface* as a whole with respect to the *object* that instantiates the *template*.

Part 3 – 7.2.2 (Interaction Rules) clearly refers to the *causality* "in the *interface's signature*", that seems to support the specification of *causalities* at the *interface signature* level. More precisely, Part 3 – 7.2.2.1 and 7.2.2.2 are clear and explicit when referring to this issue.

According to *signal interaction* rules [Part 3 – 7.2.2.1], "a computational object offering a signal interface of a given signal interface type

- initiates signals that have initiating causality in the interface's signature;
- responds to signal that have responding causality in the interface's signature."

Similarly, according to stream interaction rules [Part 3 – 7.2.2.2], "a computational object offering a stream interface

- generates flows that have producer causality in the interface's signature;
- receives flows that have consumer causality in the interface's signature."

Thus, we find that, whereas the indication of *causality* for *signal* and *stream interfaces* is defined at the *action template* level, for *operations* it is defined at the *object's interface signature* level.

One of the reasons behind these decisions seems to be the fact that, for *operations*, the *causality* indication provided by the *interface signature* determines the *causality* for each *action template* in the *interface*, depending on what we are really using: *invocations* or *terminations*. However, when dealing with *signal* or *stream interface signatures*, which comprise s*ignals* or *flows* that may go in different directions (i.e., incoming and outgoing *actions*), there is no clear relationship between the *causality* of the *interface signature*, and the *causalities* of the individual *interactions* that comprise the *interface signature*. Thus, *causalities* should be defined for each individual *action template*.

For example, let us consider a simple *stream*. Its *signature* could have both incoming and outgoing *action templates* defined for it. However, as mentioned in the *interaction rules*, we need to consider an indication of *causality* with respect to the *role* that the *computational object* plays in the communication process. This requires indicating that *causality* in the *interface signature*, which would indicate the *object* that produces the flow and the *object* that consumes it.

See [3].

Proposal

To address this issue, it is proposed that *causality* definitions should be included at both the *interface signature* and the *action template* levels. However, calling it "causality" at both levels might be confusing too. Actually, the definition of *causality* in Part 2 refers to *objects* only, i.e., the granularity of *causality* is defined at the *object* level. But in Part 3, *causality* indications seem to be used at three different levels: *object*, *interface signature* and *action template*. There is a clear need to align the granularities for these different definitions.

Thus, it is proposed to define *causalities* at every level at which this term is involved. This means incorporating *causalitie*s in individual *action templates* and in *interface signatures*. In *operations*, in which the *causality* defined at the *interface signature* level determines the *causality* of the individual *interactions*, a constraint should enforce such a relationship.

Figure 2 shows the proposal, where the indication of *causality* appears not only at the *interface signature* level—to specify the *roles* played by *computational objects* in the communication process as a whole—but also at the *interaction signature* level.

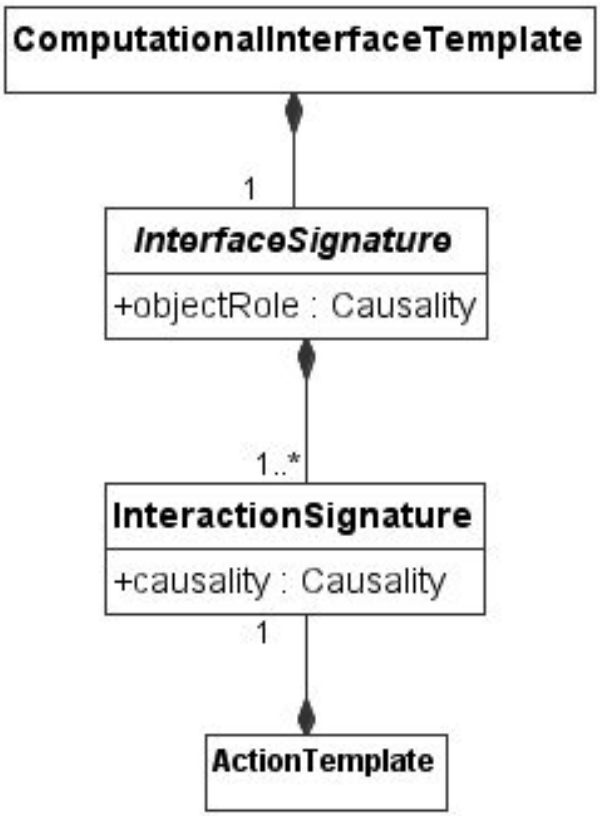

Figure 2. Indication of causality at two levels

Issue 18 Cat TR Signatures for action templates

Priority 1 – Effort: 2

Rationale

There is a problem in the way that the concept of *action template* is used for defining *operation*, *signal*, and *stream signatures*. In particular, the problem appears in the metamodel when trying to model the existing relation between *interface signatures*, *interaction signatures*, and *action templates*.

According to Part 2–9.11, a *template* is "the specification of the common features of a collection of <X>s in sufficient detail that an <X> can be instantiated using it". From this definition, it follows that an *action template* can be defined as "the specification of the common features of a collection of *actions* in sufficient detail that an action can be instantiated using it."

Action templates play a very relevant role in Part 3 in the *computational viewpoint*.

First, Part 3 – 7.1.12 indicates that "an announcement signature is an action template" (the underlining has been added). Second, a reference appears when referring to *interrogation signatures*. However, although a similar definition might be expected, Part 3 – 7.1.12 states that "an interrogation signature comprises an action template". Finally, the concept *action*

template appears again when defining *interface signatures*, which comprise a finite set of *action templates*.

So the first issue is whether *signatures* "are" *action templates*, or "comprise" *action templates*.

Furthermore, there is an issue in the way in which *action templates* are used in Part 3 for defining *signatures*. Commonly, *signatures* (of both *interactions* and *interfaces*) are considered to remain at the syntactic level, i.e., they are supposed to describe just the names and types of the *actions* and their parameters. Semantic information (e.g., *behaviour*) is not usually covered by *signatures*. However, this does not seem to be consistent with the use of *action templates* for defining *signatures*, since *action templates* might also include behavioural specifications (cf. Part 2).

Certainly, this is also corroborated by Part 4 – 4.4.2.12, which states that: "It should be noted that the text in ITU-T Rec. X.902 | ISO/IEC 10746-2 treats an interface signature as a set of action templates associated with the interactions of an interface. Given that an action template is likely to include semantic information as well as syntactic. Common interpretations of interface signature deal primarily at the syntactic level, however. […]".

See [3].

Proposal

It is proposed to solve these two issues by introducing a term that refers to the syntactic information specified by an *action template*, and to call this term "interaction signature". This term can be used to define the *signatures* of *announcements*, *interrogations*, *terminations*, *signals* and *flows* (see Figure 3), that now are "interaction signatures". This does not contradict the current standard text and, in fact, allows the separation of the syntactic and the semantic information specified by an *action template*.

Moreover, *interface signatures* (an abstract class that simply generalizes *operation*, *signal* and *stream interface signatures*) now comprise sets of "interaction signatures", which seems to be more in line with the intent of the current RM-ODP standard.

Finally, and as shown in the figure, the parameters of the *action template* are now associated to the syntactic part of such *action template*, that is, to its "interaction signature", which also seems to be more natural than attaching them directly to the *action template*.

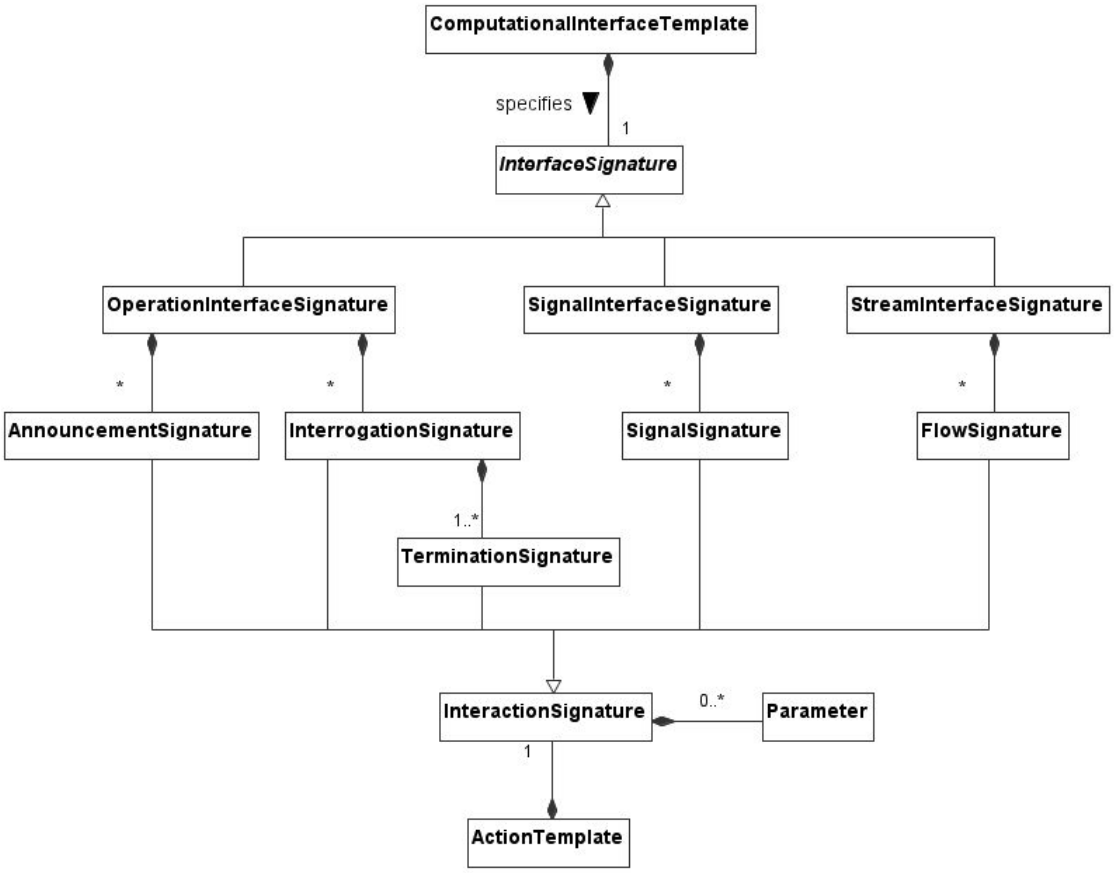

Figure 3. Interaction signatures specify the syntactic information of Action Templates

Issue 19 CL Part 3 – 7.1.5 and 7.2.2.5, Flow and use of signals to build interactions

Priority 2 – Effort: 3

Rationale

Signals are used to specify the details of flows and operations, and flows and operations are refined in terms of patterns of signals; a flow would not be represented as a composition of signals and operations.

Signals are normally introduced in the computational view in the context of interfaces to binding objects where there is a requirement to localise interfaces for the purpose of specifying QoS details.

The level of detail at which interactions are presented is a choice tbe made by the creator of the computational specification. In general, a flow is seen in the computational view as essentially unstructured. The data transfer process is seen as a single stream, with the open/close hidden in the binding process. The structure of the flow is seen in the engineering view.

However, there can be cases where the model is of the creation of a binding to support negotiation as a sequence of operations, the last of which creates a flow, followed by some

closing operation. Similarly, suspend/resume could be controlled by the application or it could be subject to a transparency.

The situation is like the distinction in Corba security between security aware and security unaware applications - the designer picks the granularity for the computational description and the transparencies, and the visibility of the detail follows from it.

Proposal

Improvements should be made to the presentation of the concept of signals and its relationship to other interaction concepts. There should be clarification of the two tier structure of interactions concepts e.g. a Note in the definition of terms and there should be clarification of the scope for designer choice.

There should also be clarification of the nature of the engineering description corresponding to a flow.

Issue 20 Part 3 computational interaction types (interactions without causalities, different patterns, etc.)

Priority 2 – Effort: 3

Rationale

The Part 2 interaction and behaviour related concepts can be used to describe kinds of interaction other than the Part 3 computational interaction types. The constraints introduced for the computational language were considered appropriate at the time. Consideration should be given to the case for relaxing the constraints and to allow some degree generalization – in particular to cover web services.

Proposal

Look at the case for relaxing the constraints and to allow some degree generalization in particular to cover web services.

Issue 21 Cat CL Explicit architectures for binding objects

Priority 1 – Effort: 2

Rationale

It would be helpful to reference ISO/IEC 14753, Interface References and Binding, which refines the concepts related to binding objects.

Proposal

Introduce a reference to ISO/IEC 14753, Interface References and Binding.

Issue 22 Cat CL Composition/decomposition of objects and interfaces

Priority 2 – Effort: 5

Rationale

There is a need to clarify the nature of *object* and *interface composition* and *decomposition.*

Proposal

Provide clarification of the nature of *object* and *interface composition* and *decomposition.*

Issue 23 Cat TR There is a need to clarify the nature of correspondences that apply between viewpoints.

Priority: 2 – Effort: 7

Rationale

The current text on the nature of correspondences between viewpoints does not provide sufficient direction for the development of specification and needs further refinement and clarification

Proposal

It is currently unclear how to resolve this issue.

Issue 24 Cat CL There is a need to clarify the nature of the relationship between computational and engineering viewpoints

Priority: 2 – Effort: 4

Rationale

There is a need to address the following issues:

- Interface, Signature, and Interaction: How can one provide in an engineering specification elements corresponding to these aspects of computational objects? (e.g. how is it possible to specify "interacting engineering objects" without having these concepts from the Computational viewpoint?) Is there any inheritance rule from Computational viewpoint language to Engineering viewpoint language?
- Selective transparency: Part 3 clause 16 says "Distribution transparency is selective in ODP system." How should we draw a line between Computational and Engineering if only a portion of distributed transparency were selected (e.g. if a Computational viewpoint specification only assumes access transparency, failure transparency and location transparency and does not assume other transparencies? Does one need to construct an engineering viewpoint type of specification within the computational specification?)
- Selected use of ODP function: Part 3 clause 16 further says "Transparencies are defined as constraints on the mapping from a computational specification containing a transparency schema to a specification that uses specific ODP functions and engineering structures to provide the required form of masking." This is like saying

that ODP functions work as "transparency mapping pattern" or "transparency requirements" when Computational specification is transformed into Engineering specification, and still is selective. If no ODP function were selected, what would be the difference between a computational specification and the corresponding engineering specification (especially a difference between specification of computational objects and of basic engineering objects)? A related issue is the question of which viewpoint provides a specification of the use of selected ODP functions (e.g. computational only?).

Proposal

Provide clarification of the issues raised in the rationale.

Issue 25 Cat TR There is a need to clarify the nature of the technology viewpoint and its difference from the engineering viewpoint

Priority: 1 – Effort: 4

Rationale

The concepts for a metamodel for the technology viewpoint are not obvious and the concepts currently in the standard are minimal. It necessary to have something more and to clarify the difference between the technology viewpoint and its difference from the engineering viewpoint

Proposal

Provide a more adequate set of concepts for a metamodel for the technology viewpoint and clarify its difference from the engineering viewpoint.

Issue 26 Cat TR Infrastructure and abstract platforms

Priority 2 – Effort: 3-4

Rationale

In [19] Blair and Stefani have equated the boundary between the *computational* and the *engineering viewpoints* to the distinction between application and infrastructure: "It is important to realize that the boundary between the two *viewpoints* is fluid, depending on the level of the virtual machine offered by the system's infrastructure. Some systems will provide a rich and abstract set of *engineering objects* whereas others will provide a more minimal set of objects leaving more responsibility to the applications developer."

According to this interpretation, specifications in the *computational viewpoint* are influenced by the level of support provided by the infrastructure. By setting the level of support provided by the infrastructure, one can refer to *computational* concerns and *engineering* concerns. Equating infrastructure to predefined middleware platforms would lead us to the conclusion that *computational specifications* are directly influenced by the level of support provided by a selected middleware platform. *Computational specifications* would therefore be, to some extent, platform-specific. In this case, the separation of *computational* and *engineering* concerns would be identical to the separation between application and middleware platform concerns. The reusability of a *computational viewpoint* specification would be restricted by

its dependence on platform characteristics. Furthermore, from the perspective of application developers, the separation of *computational* and *engineering* concerns would be implied by the availability of a software infrastructure. Therefore, we conclude that the motivation for the separation of *computational* and *engineering* concerns is predominantly bottom-up.

Another interpretation for the infrastructure assumed by the *computational viewpoint* is that of an 'ideal infrastructure'. In this interpretation, the motivation for the separation of *computational* and *engineering* concerns is predominantly based on the needs of the developer to handle the complexity of application and infrastructure separately, regardless of the availability of a software infrastructure. The *engineering viewpoint* offers the possibility for a designer to engineer the infrastructure explicitly. While this interpretation is ideal from the perspective of separation of concerns for the application developer, it does not leverage the reuse of middleware platforms, which would significantly improve the efficiency of the development process.

Committing to one of theses discussed interpretations of infrastructure is undesirable for the adoption of *computational viewpoint* concepts in a model driven approach to system specification, since it may lead to models at a low level of platform independence, or it may lead to models that cannot be realized on existing middleware platforms.

See [4] and [8].

Proposal

Equate the term infrastructure, as used in RM-ODP, to the notion of abstract platform.

An abstract platform is defined in terms of the *bindings* supported, the *transparencies* supported, and the types of *quality-of-service* (QoS) constraints that may be applied to interface contracts. The use of *binding objects* may provide considerable flexibility to implementations of platform-independent models, since it is possible to provide countless different implementations of a *binding object*. In addition, there is considerable freedom in choosing mechanisms for obtaining a required t*ransparency* and satisfying QoS constraints. At any point in a design trajectory, a mapping to a platform-specific realization may be defined, as long as: (i) the semantics for the original model are respected, as defined by the vocabulary and rules of the *computational language*; and (ii) quality characteristics of the realizations obtained through mappings are acceptable.

This approach can be beneficial for the development of distributed applications, so that a proper balance can be obtained between the following design goals:

- designers can use the separation of application and infrastructure concerns to cope with the complexity of distributed application design;
- middleware platforms can be reused to improve significantly the efficiency of distributed application development; and
- platform-independence can be obtained as a means to preserve investments in application development and withstand changes in technology.

A consequence of equating infrastructure to abstract platform is that *computational viewpoint* concepts can be applied recursively at different levels of platform independence. The use of the same conceptual framework for different levels of platform-independence facilitates the definition of correctness relations or even automated transformations.

Issue 27 Cat TR Legacy systems

Priority: 3 – Effort: 7

Rational

There is a need for guidance on the relevance of RM-ODP concepts and approaches in the specification of migration from legacy systems

See [7].

Proposal

It is currently unclear how this issue can be addressed.

Issue 28 Cat CL/TR ODP Guide – RM-ODP Part 1

Priority 2 – Effort: 10

Rationale

There is a need to revise RM-ODP Part 1:

1. to reflect any revision of Parts 2 and 3;

2. to provide better examples or replace the present examples by a reference to the example under development in the work on ISO/IEC 19793.

3. to explain how RM-ODP concepts are aligned with, or relate to, other standards and publicly available specifications such as OMG's MDA, CORBA and Action Semantics, and EJB.

Proposal

Revise RM-ODP Part 1 to cover the issues identified in the Rationale.

References

NOTE – Papers from WODPEC 2004 are available at
http://www.lcc.uma.es/~av/wodpec2004/WODPEC2004-Proceedings.pdf

[1] H.Kilov, Semantic interoperability: using RM-ODP to bridge communication gaps between stakeholders. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[2] P.F. Linington, What Foundations does the RM-ODP Need? In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004

[3] R. Romero and A. Vallecillo, Action Templates and Causalities in the ODP Computational Viewpoint. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[4] J.P. Almeida, M. van Sinderen and L. Ferreira Pires, The role of the RM-ODP Computational Viewpoint Concepts in the MDA approach. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[5] Y. Nagase, D. Hashimoto and M. Sato, Applying Model-Driven Development to Business Systems using RM-ODP and EDOC. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[6] A.P. Gonçalves Serra, S.A. Vicente, D. Karam Jr and M. Martucci Jr, Architecting frameworks for specific applications with RM-ODP. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[7] R. Le Delliou, N. Ploquin, M. Belaunde, R. Bendraou and L. Féraud, A Model-Driven Approach for Information System Migration. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[8] Lea Kutvonen. Challenges for ODP-based infrastructure for managing dynamic B2B networks. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[9] D. Akehurst, Proposal for a Model Driven Approach to Creating a Tool to Support the RM-ODP. In *WODPEC 2004 -- Workshop Papers* from *EDOC 2004*, Monterey, California, September 2004.

[10] P. F. Linington and W. F. Frank, Specification and implementation in ODP, In J. Cordeiro and H. Kilov, editors, *Proceedings of the 1st Workshop on Open Distributed Processing: Enterprise, Computation, Knowledge, Engineering and Realisation, pages 69-80,* Setubal, Portugal, July 2001. ICEIS Press.

[11] P. F. Linington and S. Neal, Using policies in the checking of business to business contracts, In H. Lutfiyya, J. Moffat, and F. Garcia, editors, *Fourth IEEE International Workshop on Policies for Distributed Systems and Networks, pages 207-218*, Lake Como, Italy, June 2003. IEEE Computer

[12] P. F. Linington, Z. Milosevic and K. Raymond, *Policies in Communities: Extending the ODP Enterprise Viewpoint*, In *Proceedings of 2nd International Workshop on Enterprise Distributed Object Computing (EDOC98)*, San Diego, USA, November 1998.

[13] ITU-T Recommendation X.901 | ISO/IEC IS 10746-1, Information technology - Open Distributed Processing – Reference Model: Overview, 1996.

[14] ITU-T Recommendation X.902 | ISO/IEC IS 10746-2, Information technology - Open Distributed Processing – Reference Model: Foundations, 1996.

[15] ITU-T Recommendation X.903 | ISO/IEC IS 10746-3, Information technology - Open Distributed Processing – Reference Model: Architecture, 1996

[16] ITU-T Recommendation X.904 | ISO/IEC IS 10746-4, Information technology - Open Distributed Processing – Reference Model: Architectural Semantics, 1996.

[17] ITU-T Recommendation X.911 | ISO\IEC 15414, Information technology - Open Distributed Processing-Enterprise Language, 2001

[18] ISO/IEC IS 10165-7, Information Technology – Open Systems Interconnection – Structure of management information: General relationship model, 1996.

[19] G. Blair and J.B. Stefani, Open Distributed Processing and Multimedia. Addison Wesley, 1997

[20] ITU-T Rec. X.952 | ISO/IEC 13235, Information technology - Open Distributed Processing - ODP Trading Function

[21] G.E.Hughes and M. J. Cresswell, A Companion to Modal Logic, Methuen, London, 1984

[22] SC7/WG19 BRI-015, Working Draft for ITU-T Recommendation X.906 | ISO/IEC 19793, Information technology – Open distributed processing — Use of UML for ODP system specifications