Using the ODP reference model for Enterprise Architecture

Lea Kutvonen
Department of Computer Science, University of Helsinki, Finland
Lea.Kutvonen@cs.helsinki.fi

Abstract

The Open Distributed Processing Reference Model (ODP-RM) provides viewpoints and abstract infrastructure guidelines that can be used for a basis for enterprise architecture, especially for an inter-enterprise architecture. The ODP-RM does not prescribe methodology for modeling itself, but provides common vocabulary and focus for description. This paper performs a brief analysis of the ODP-RM (and Pilarcos extensions to it) in terms of defining enterprise architecture. Special attention is given to potential to provide support for consistency enforcing between viewpoints, analysis of model properties, and interaction between business and technology needs.

1. Introduction

"An enterprise architecture describes how the elements of an organization fit together – the business processes, organizations responsible for them, Information Technology (IT) capabilities and infrastructure – today and in the future" [1]. For this purpose, a large number enterprise architecture methods, architecture frameworks and modelling tools already exists (e.g., Zachman [2], TOGAF [3], ARIS [4], Archimate [5], business architecture [6]) as well as standard frameworks [7]. By their origin, the main goal has been to enforce a shared vision of the enterprise architecture to all involved parties (users, designers, implementors, service providers) in the enterprise, and to support coherent management even during changes in the architecture.

Modern challenges for enterprise architecture arise from the interoperability and federation needs: inter-enterprise computing and merging of businesses cause needs for loose integration of business services across enterprise boundaries. Furthermore, there is increasing demand for aligning the computing infrastructure to the requirements of the business management activities. Interoperability and federated management facilities become fundamental requirements for the collaborative architectures, covering processes, information, and modularization of the application domain.

The Open Distributed Processing Reference Model, ODP-RM [8,9], was initially developed for structuring the application level concerns of communicating systems in general. As part of the present trend, business networks form an important focus area for which the reference model can be applied. This paper discusses the above mentioned enterprise architecture challenges and required elements in the enterprise architecture and points out what kind of solutions and architecture patterns are already present in the ODP-RM. As refinements of the ODP-RM generic specifications, results from the Pilarcos project work are used. The Pilarcos project has developed B2B middleware [10] and is enhancing the research activities towards service oriented software engineering and enterprise architecture needs in a way consistent with the ODP reference model.

Section 2 discusses formation of enterprise architecture in general, while Section 3 states some guidelines for a specific enterprise architecture for networked business arena. This inter-enterprise architecture is made visible through papers on the Pilarcos work [1], which is shortly commented on in Section 4 while the main contents of Section 4 is on noting the solutions suitable for inter-enterprise architecture directly provided by the ODP-RM. Section 4 further elaborates on the potential for cross-viewpoint consistency maintenance both at architecting time and operational time through monitoring and operational knowledge collection and feedback processes.

2. Defining enterprise architecture

An enterprise architecture defines how business strategy is realized using the processes and resources available by the enterprise. The process by which the architecture is defined and that evaluates its change needs and describes its development steps rely on a) common terminology, b) the methodology for

developing the system and its components, c) explanation on how the developed components fit together, and d) the set of tools that support the methodology.

The enterprise architecture itself is commonly divided to topic domains and layers of different levels of detail. The domains should become connected together by methodological means that supports consistency and refinement steps. The variety and maturity of tools for model creation and verification preserves attention too.

The subarchitectures commonly in use include [e.g.,3] Business Architecture, Data Architecture, Applications Architecture, and Technology Architecture.

The business architecture is needed to start the definition process, to define the business strategy, the business processes of interest, and to define the stakeholders and their viewpoints of interest to be followed through in the other architecture models. The resulting documentation should point out the reference framework to be used, and identify the architecture principles to be used in the architecture process. Suitable reference architectures for business architecture could include RosettaNet specifications etc.

The data architecture and application architecture have the goals of defining the major types and sources of information in a consistent, complete and stable manner so it can be accessed and understood by all stakeholders. On application side, the objective is to determine the selection of information processing applications relevant for the enterprise.

The technology architecture grounds the work on an existing application platform, shared terminology and production and deployment processes. Thus the selected technology architecture restricts the correspondences between the business and application architectures and the technology support available or required.

The impact of the existence, maturity and timeliness of the enterprise architecture is significant: it allows informed decision-making while negotiating commitments on inter-enterprise collaboration, on provision of products or services to clients, and controlling changes in the enterprise architecture. When interoperability and collaboration management are specifically addressed, the enterprise architecture also has a considerable impact on the enterprise's agility.

3. Architecting networked businesses

Loosely-coupled, dynamic collaborations of business services that are provided by autonomous enterprises form a specific category of virtual enterprises (or extended enterprises, business networks, etc.). This section summarizes a few enterprise architecture guidelines for enterprises wishing to emphasis potential for agility in business networks.

In this future vision of inter-enterprise architecture, the essential concepts are business service interoperability and contract-governed collaboration between business services.

We understand interoperability, or the capability to collaborate, as the effective capability to mutually communicate information in order to exchange proposals, requests, results, and commitments (i.e., to exchange speech acts common in business). The term covers technical, semantic and pragmatic interoperability. Technical interoperability is concerned with connectivity between the computational services, allowing messages to be transported from one application to another. Semantic interoperability means that the message content becomes understood in the same way by the senders and the receivers. This concerns both information representation and messaging sequences. (More and more the ability to transform information in messages between the source and the targets is viewed as a service supporting interoperability (e.g., enterprise application integration (EAI) and enterprise service bus (ESB) [11]). Pragmatic interoperability captures the willingness of partners to perform the actions needed for the collaboration. This willingness to participate refers both to the capability of performing a requested action, and to policies dictating whether it is preferable for the enterprise. (In most integration products, workflows are used to manage business processes. Here, the joint understanding of external business processes and their enactment is essential, not the actual enactment technology.)

We expect the technology architecture to provide common facilities for interoperability and collaboration lifecycle management. The method of managing collaborations and contracts is based on generic, B2B middleware protocols and agents private to each involved enterprise. In addition to the common protocols, each enterprise needs expert systems to support private decision-making on participation in collaborations, management of private policies, etc.

The methodology for building the business networks is semi-automated: based on a selected business network model and service offers published by service providing enterprises the B2B middleware is able to suggest contracts that are ensured to represent interoperable collaborations. The expected way of

business services to fit into the collaboration and to each other is defined in terms of interoperability and collaboration contract requirements and breaches. These aspects are to be continuously monitored during the collaboration lifecycle, at times triggering management actions.

We call all kinds of the inter-enterprise collaborations business networks. This is because the business management activities appearing in different types of business scenarios (e.g., supply chains, virtual enterprises, and subcontractor networks) repeat largely the same pattern. Looking at the common business scenarios from the supporting technology point of view, we can separate external business processes that express what interactions the players in the business network must take, and the service processing software at the location of each player. The nature of supply chain or virtual enterprise becomes expressed and defined by the business processes, while the supporting technical environment can be identical for all types. From the technical point, we can consider that the primary goal of each independent organization in these scenarios is to provide added-value services by composing existing services provided by different enterprises. However, because of the different responsibility models important for the business management perspective we need to preserve the following separation:

- orchestration, where the coordinator of the composed service takes on the obligations of providing the service; and
- collaboration, where a mutual contract is formed and the members of the collaboration are equal and have their contracted obligations; the coordination of the collaboration is maintained by the supporting infrastructure.

Therefore, each business network is viewed as collaboration between autonomously administered business services. A business network is established dynamically to serve a certain business scenario or opportunity that is made commonly known by publishing a business network model (BNM). The business network model captures all those external business processes that are relevant for the business scenario. The business network model also gives structure for the contract that is technically used for governing the collaboration at runtime; the contract captures most of the social behaviour requirements in the collaboration. A business service is realized by a business application implementation running under the administration of a single authority. The potential of activities of the business application is restricted and controlled by enterprise policies to the degree that the enterprise is prepared to make available for its clients. The business services can fulfill roles in multiple business networks simultaneously, based on their ability to fulfill the behavioural and nonfunctional requirements of the role and the contract.

In terms of defining an enterprise architecture for networked business, we can state the following guidelines. First, business scenarios should be formed in a rather generic way and be refinable by policy statements that can be negotiated for each collaboration separately. The business scenarios should be modelled in terms of business processes, compositions of business processes, and provide control and monitoring potential through addition of non-functional properties. The resulting models should be verified and published on trusted repositories available for all stakeholders.

Second, the open market of business services is formed by federated service offer repositories, that are trusted in terms of providing non-repudiation of offers and traceabilty of the offer providers, and in terms of checking conformance to known service types and associated required properties within offers.

Third, business services are mapped to technical service compositions under a single business administration that has authority on deciding the associated policies and contracts that constraint the behaviour of the technical service. This authority must be authorised and responsible for the enterprise decision-making and traceable.

Fourth, the application platforms involved should be able to support a service-oriented architecture. This means that there should be either native or added-on facilities to provide metainformation on the services and reflect the metalevel commitments to the system state (or to refuse changes), and strict encapsulation of the implementation of the services provided through the published interfaces.

Fifth, the technology environment should support not only concepts for publishing, binding and performing services, but to manage the lifecycle of collaborations and to detect and maintain potential for interoperability as needed.

Sixth, the technology architecture is a changing element. Therefore, an especially important principle to cover in the architecture process is to define solutions that cut potential domino-effects from technology changes to directly supported business or application chains forcing re-implementations; legacy support, or rather strong encapsulation of services (not only objects or components) is a necessity.

Finally, the technology architecture should support global description of relevant information between

services, but also relevant metainformation about the services and collaborations.

The difference to other approaches can be found in the strong encapsulation of business services, deliberate omission of distributed execution facilities for shared business processes, and emphasis on the ability of the architecture itself to carry out the changes in the business architecture, information system architecture, and technology architecture. For all these aspects, the B2B middleware is expected to provide semantically consistent repositories and associated protocols for publishing, retrieving, comparing and negotiating. In addition, the view of the architecture is global, instead of a single enterprise.

4. The ODP-RM solutions

In the early phases of the Pilarcos project series, involvement on the ODP standardization was part of the work. Therefore, a lot of alignment of the basic concepts can be easily found. In this section, a brief introduction to the ODP-RM is given, pointing out aspects on which we can base the future, inter-enterprise architecture view.

4.1. Family of ODP standards

The ODP reference model (RM-ODP) [8,9,13,14, 15] is a joint standardisation effort of ISO and ITU. It was started with a basic reference model standardisation officially already in 1989, and developed in interaction with other distributed system models. Thus, ODP reference model has had impact on industry trends already during its development.

The ODP standardisation aims for development of standards that allow distributed information processing systems to be exploited in a heterogeneous environment and under multiple organisational domains . This goal is an enhancement to the openness requirement above. In addition to the use of public and standardised middleware services, the systems must be able to support interoperation in spite the independent evolution and independent technology decisions made by the sovereign member systems.

The ODP standards support systems to be built so that they a) provide software portability and interoperability; b) support integration of various systems with different architectures and resources without costly ad-hoc solutions; c) accommodate system evolution and run-time changes; d) federate across autonomously administered or technically differing domains; e) incorporate quality of service aspects to failure concepts; f) include security service; and g) offer selectable distribution transparency services for communication.

These goals are acquired by the ODP reference model through three already standardised aspects of the basic reference model (which that was completed in 1996 [8,9]): First, a division of an ODP system specification into viewpoints, in order to simplify the description of complex systems [9]. Second, definition of a set of general concepts for expressing the viewpoint specifications [8]. Third, a model for an infrastructure supporting, through the provision of distribution transparencies, the general concepts that it offers for specification purposes [9]. Finally, specification of some essential middleware services as component standards. These services include the already completed trading service [15], naming framework [16], type repository function [17], and interface binding framework together with the supporting protocols.

4.2. Common concepts

Part 2 of the reference model provides a common vocabulary to be used in any ODP system specifications.

Presently, the ODP reference model is often undervalued solely due to its object-orientation. However, the objects used are only a form of abstraction, of strong encapsulation of behaviour and information and more like the present service concepts. In the correction cycle of the ISO and ITU standards, it is suggested to point out the relationship of the present object-oriented vocabulary and more modern service concepts; the relationship is fairly straightforward [18].

One of the necessary steps for managing collaborations across administrative domains is to differentiate between ways by which objects (i.e., services) appear in the system. The ODP model allows objects to appear in the system in two ways, by instantiation or by introduction. Instantiation is the often used model, where a template exists with sufficient information for the creation of an object instance. Introduction allows manipulation of objects without knowledge of their instantiation methods – only the object type is interesting. The ODP model defines type as a predicate that classifies objects based on their properties. A template is defined as a type detailed enough for instantiation. The instantiation process is naturally dependent on the platform facilities, and therefore, whether a type is also a template depends on the platform.

The object behaviour is specified as a set of interfaces. An interface represents objects role as a

provider or exploiter of a service. As an object can support multiple interfaces, it can also participate in the provision of multiple services. A typical object supports at least a mission specific interface and a management interface.

An interface (at which two or more objects meet for communication) is not specified as an indivisible, global abstraction. Instead, both a client requesting a service and a server providing such a service, can have slightly different technical views of the interface. The ODP communication model concepts declare how these views can be mapped together in the binding process that creates a communication channel between the client and server interfaces. Object interfaces are bound based on their type, the object template is not considered. Because of the separation of client and server interfaces, also the sub-typing and substitutability concepts for ODP objects differ from other object models. The type system of objects focus on the shared features required from interfaces that need to be bound together, instead of focusing on implementation inheritance hierarchies. The essence of sub-typing rules for ODP objects is contravariance: the offered information must include at least the information expected by the receiver. In most object models, sub-typing is based on covariance: the replacing object can both expect to receive and offer more information that the replaced object expects and offers.

The separation of interfaces allows not only late binding across enterprise boundaries, but also for relaxed type-matching and automation of transformation configurations into the communication channels [19].

The ODP reference model introduces the structuring concepts of community, domain, and federation. These concepts can be used for organising objects for producing and exploiting services. The structuring concepts can be considered to be either static, design time concepts, or dynamic, operation time concepts.

A community is a configuration of objects with a common objective. For example, a business network is a community where the common objective is explicitly stated in the governing contract, capturing joint business processes and policies as means to reach the objective.

A <X> federation is community of <X> domains. A <X> domain is a set of objects, each of which is related by a characterising relationship <X> to a controlling object. For example, a technology domain is the set of objects conforming to a technical standard, and a trading domain is the set of objects known to a trader object. An example of a federation is a business network where a group of B2B middleware agents work together to manage the collaboration lifecycle and interoperability.

4.3. Viewpoints supporting architecture domains

The ODP reference model defines five viewpoints – enterprise, information, computational, engineering and technology viewpoints [9]. The viewpoint specifications of a system can be regarded as separate projections of a full system description. A system must be specified from each of the viewpoints. Each viewpoint specification is a consistent and complete specification on its own, but it only considers those aspects of the system that are valid on its point of view. So the viewpoint specifications do not overlap totally, but they may show different level of detail in the areas where they need to discuss same or related features. The engineering viewpoint specifications are tightly related with the ODP infrastructure model that is specified as part of the engineering viewpoint specification rules.

The enterprise viewpoint description of a system specifies the activities and the responsibilities of the system. Activity means any information exchange sequence and it is a high-level abstraction of the operations within the system. The system itself can have any granularity that is interesting. The system can be as wide as a global information network including applications or as small as a memory cache. The enterprise specification identifies the system, its environment, and the required communication of the system and its environment. The specification answers to the questions “What is the purpose of the system?” and “What services the system is responsible to provide?” and “Who needs the services?”.

In respect of the enterprise architecture, each enterprise viewpoint specification provides provide a business architecture model, although without any connections to component services to realise the architecture or to technology architecture to support it. However, the business network models that follow rather closely the enterprise language structures and requirements [20] provides definitions for expected business values, and other non-functional aspects. The enterprise specification thus sets guidelines that will be reflected to the runtime environment, to the monitors governing the collaboration and the business services; the linkage between business architecture and technology architecture is direct.

The information viewpoint description of a system identifies logical information entities, their logical

contents, their repositories and the objects that are responsible of the information flow in the systems. Questions for information viewpoint specification are “What information is needed to support the system's services?”, “Where does the information come from and go to?”, and “Is it necessary to store the information somewhere?”. The information viewpoint specifications should not describe data structures, but only the semantics of the information. Also, the technique of storing information is irrelevant in this viewpoint (as the logical infrastructure supports storage services).

In respect of the enterprise architecture, each information viewpoint specification provides a set of relevant information elements that can be published in common type repositories, making the information accessible to all involved stakeholders. However, associating the information definitions to business network models, a more effective and non-overlapping information base can be formed [22].

The computational viewpoint specification captures the behaviour of the system. Behaviour is an abstraction of how things are done, in contrast to the notion of what things are characteristic in enterprise viewpoint activities. An activity identified in enterprise viewpoint may involve several objects to perform a sequence of operations in computational viewpoint. The computational viewpoint shows the system as a composition of logical objects. For each object its interfaces are described. If the interface involves operations, each operation gets logical parameter descriptions (information structures, not data structures) – if the interface involves streams, each data flow component of a stream gets logical protocol descriptions instead. This is the viewpoint that usually explicitly shows potential for distribution. Neither the enterprise viewpoint nor the information viewpoint specifications need to express any distribution concerns. The computational viewpoint answers to questions like “Which operations are available?”, and “Who (which logical entity) performs the operation?”.

In respect of the enterprise architecture, the computational viewpoint specifications can be written so that they provide structure for business services and thus elements to publish on the trading service to form a global open service market. It is essential that there is service types publicly defined in the type repository to provide for mapping between business network roles and business services.

The engineering viewpoint specification identifies the infrastructure services needed for the system to operate. The ODP-RM engineering viewpoint defines the set of available infrastructure services, and all other engineering viewpoint specifications should show how the specified system utilise these services. The engineering specification therefore answers the question “By which services are the computational objects supported?” The ODP infrastructure model identifies a set of global, distributed basic services that should be available at each node in the global system. These include invocation of operations, transfer of continuous data as streams, trading, type repository functions, etc. These services facilitate selective transparency of communication.

In respect to the enterprise architecture, the engineering viewpoint specifications should capture the functionality of the support environment as enhanced with the B2B middleware repositories, especially their contents, and the associated protocols between the agents representing the enterprises. These elements contribute to the technology architecture, and in addition importantly to the business architecture, information architecture and application (service) architecture.

The technology viewpoint specification shows in a concrete hardware and software configuration how the system services and other required components are realized. The specification answers the question “How are the infrastructure services realized?”. This is a direct mapping to the technology architecture.

The ODP viewpoint languages or specifications address the required elements for enterprise architecture definition, as can be seen for example by reviewing the TOGAF architecture development method. In the comparison, it should be noted that the TOGAF architectures is a partitioning of the set of models required to do enterprise architecture, not a set of overlapping views. TOGAF also recognises the need for views, that is an assembly of basic architecture elements for a purpose.

Also, it should be noted that each viewpoint specification adds to the set of further requirements on the structure of service realisations, service descriptions and service management information. The method that can be built using the ODP-RM is thus incremental; when the viewpoint specifications are made available through public repositories this means a globally evolvable collaboration knowledge base and automated breeding environment. In addition, different enterprises may have a different set of coexisting specifications in use, making the environment flexible for different types of enterprises or different maturity levels of collaboration. The management structure is described in the following section.

4.4 ODP-RM enhancements by B2B middleware

The ODP reference model defines an abstract computing platform that comprises of a set of coordination, management, and repository functions. Each open system should independently support these functions, and in some restricted cases, also allow other systems to make controlled queries on the supported interfaces. The functions are described using a set of supporting concepts (nodes, capsules and clusters), that give an internal engineering view of the middleware software. However, these concepts are used only for descriptive purposes, to ease discussion, not as technology guidelines.

At the time of the ODP-RM creation, the abstraction level for distribution platforms required that kind of structuring, and still does, as the platforms have not yet developed to level of maturity where there were a consistent set of concepts available for selecting properties like security, privacy, or dependability for the transparent configuration of the communication layer.

However, on top of this communication layer, more conceptual functionality must be provided. The essential functions introduced in the ODP-RM include trading function and type repository: The trader provides a service offer repository [15], while the type repository [17] provides structuring rules and constraints for metainformation and service elements managed by the platform.

In the Pilarcos environment we have extended this abstract platform by management services, i.e. pervasive functions as follows. First, tools and repositories support developing and publishing of new models for business networks,and defining new service types for business services in such a way that the service types match the needs of the business network roles [21]. Second, service offer repositories enable enterprises to publish business services to the open service markets together with metainformation for automated matching to roles and for interoperability testing against peers in the business network [22]. Third, means are required for declaring policies that govern the use and the availability of business services. Fourth, new protocols are needed for negotiating eContracts to govern a new business network [23]; the establishment phase is partially performed by a third-party population process, partially by a collective, refining or dropping-out negotiation protocol between becoming peers. Finally, facilities are needed for monitoring the behaviour within eCommunities and manage breaches within them as specified in the eContract [24]. We believe that by this kind of generic B2B middleware services that are available through private agents at each enterprise, the right kind of software investment cycles can be supported.

4.5. Consistency of viewpoint models

The system specification includes five complete specifications, that all can be analysed as separate. Each viewpoint reveals a different aspect of the system, and therefore the full functionality can only be seen by looking at all specifications together. As the viewpoint specifications are all complete specifications on their own, the abstraction levels of objects in the specifications can differ. Still, the specifier must show how the specifications are mapped together.

The ODP-RM aims for specifications that allow software portability and interoperability and defines four classes of reference points – points where the conformance to the standard specification need and is allowed to be tested. Other behaviour than that visible at these reference points is not considered. The four classes of reference points are programmatic, perceptual, interworking and interchange conformance points. For service-oriented architecture these two are of interest: at a perceptual reference point there is some interaction between the system and the physical world while at an interworking reference point sets conformance requirements for communication between two or more systems in terms of the exchange of information.

The conformance rules can be modeled across any viewpoint specifications. For example, conformance rules can expect enterprise viewpoint policies to be adhered to at the technology level, or the information exchange patters published to be followed at all business processes and business services playing roles in them.

In respect of enterprise architectures, the conformance rules would be captured in methodology guidelines.

In terms of consistency of the models, we need to use the facilities provided by the viewpoint languages themselves. Because the enterprise architecture clearly utilizes a reflective system design, the semantic constraints of metainformation is a key issue. Here, the ODP information viewpoint language gives a good basis by stating static schema, dynamic schema and invariant schema, thus denoting the initial state of the system, allowed state changes and their effect on information content, and semantic restrictions that always hold over the information state.

By using the information viewpoint over the B2B middleware repositories, we can define semantic constraints between newly defined and published service types, business network models, and service offers. When the repositories each have a set of required information contents defined according to the B2B management protocol needs, the essential type safety of the overall system can be maintained [25]. This promises for more effective and global tools than is traditionally expected by enterprise architecture methods for single enterprises.

5. Conclusion

We presented an enterprise architecture for networked business. Each involved enterprise need to have their private architectures, but in addition be able to participate this larger architecture as a member in the collaboration management activities.

The ODP-RM provides sufficient concepts to build an evolvable enterprise architecture of this type. Although some of its vocabulary needs bringing up to date by adoption of service-oriented concepts and enhancing discussion on non-functional properties, the framework is still valid and in line with current widely used description languages.

In comparison to traditional enterprise architecture methods, the evolution steps of the architecture have been taken as a part of the computing system responsibilities and loosely-coupled interoperation that can be modified and reconfigured at operational time as a main method of collaboration.

References

[1] Mitre, “Guide to the Enterprise Architecture Body of Knowledge”. http://www.mitre.org/tech/eabok/, 2004.

[2] J. A. Zachman, “A framework for information system architecture,” IBM Systems Journal, 26:3, pp. 276–292, 1987.

[3] The Open Group, “Togaf 8.1.1 online,” Tech. Rep., http://www.opengroup.org/architecture/togaf8-doc/arch/.

[4] A.-W. Scheer and M. Nüttgens, “Aris architecture and reference models for business process management,” Tech. Rep., 1996.

[5] M. Lankhorst and H. van Drunen, “Enterprise architecture development and modelling -- Combining TOGAF and ArchiMate” Via Nova Architectura. March 2007.

[6] G. Versteeg and H. Bouwman. Business architecture: a new paradigm to relate e-business strategy to ICT. ECIS 2004.

[7] K. Kosanke, “Standardization in enterprise inter- and intraorganizational integration,” International Journal on IT Standards and Standardization Res., 3 (2), pp. 42–50, 2005.

[8]. Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 2: Foundations, 1996. IS10746-2.

[9] Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 3: Architecture, 1996. IS10746-3.

[10] L. Kutvonen, J. Metso, and S. Ruohomaa, “From trading to eCommunity management: Responding to social and contractual challenges” ISF 9(2-3):181-194.

[11] D. A. Chappell, Enterprise Service Bus. O’Reilly. 2004.

[13] Linington, P. RM-ODP: The architecture. In The 3rd International Conference on Open Distributed Processing – Experiences with distributed environments, 1995.

[14] Raymond, K. Reference model of open distributed processing (RM-ODP): Introduction. In The 3rd International Conference on Open Distributed Processing – Experiences with distributed environments , 1995.

[15] Blair, G., and Stefani, J.-B. Open Distributed Processing and Multimedia. Addison-Wesley Publishing Company, 1997.

[16] Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – ODP Naming framework. IS14771.

[17] Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. ODP Type repository function. IS14746.

[18] Lea Kutvonen and Janne Metso. Services, contracts, policies and eCommunities - Relationship to ODP framework. WODPEC 2005, pages 62-69, 2005.

[19] Lea Kutvonen. Relaxed Service-type matching and Transformation management. In Workshop on Enterprise Modelling and Ontologies for Interoperability, EMOI-INTEROP 2005, June 2005.

[20] Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – RM-ODP enterprise language. 15414

[21] T. Ruokolainen and L. Kutvonen. Service typing in Collaborative Systems. In Interoperability for Enterprise Software and Applications Conference (I-ESA2006).

[22] L. Kutvonen, T. Ruokolainen, and J. Metso. Interoperability middleware for federated business services in web-Pilarcos. IJEIS 3:1, 1-21.

[23] L. Kutvonen, J. Metso, and T. Ruokolainen. Inter-enterprise collaboration management in dynamic business networks. In On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE, Agia

[24] J. Metso and L. Kutvonen. Managing Virtual Organizations with Contracts. In Workshop on Contract Architectures and Languages (CoALa2005), Enschede, The Netherlands, Sept. 2005.

[25] Lea Kutvonen. What applying of the ODP viewpoints teaches us about tool-chains. In 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06), 2006. IEEE Computer Society.