Challenges for ODP-based infrastructure for managing dynamic B2B networks

Lea Kutvonen
Department of Computer Science, University of Helsinki
Lea.Kutvonen@cs.Helsinki.FI

Abstract

The availability of open networks and the rise of service-oriented architectures have created an environment where collaboration between enterprise ICT systems becomes technically plausible. The current challenges for collaboration management focus on ensuring the semantics and pragmatics of collaborations. It is especially interesting to capture the inter-enterprise business processes in such a way that autonomous computing systems can control and manage collaborations of that form. Furthermore, management of open collaborations requires shared concepts and protocols for trust management.

The reference model of open distributed processing (RM-ODP) standards deal with distributed information processing systems that are exploited in a heterogeneous environment, under multiple organizational domains. The standards provide, besides general terminology and viewpoints for division of system specifications, a model for an infrastructure that supports distribution transparent communication at application level.

Our contributions to the field involve a B2B middleware architecture for managing application level collaborations in a evolvable and dynamic way. The work is consistent with RM-ODP, but extends the management and communication facilities. This paper discusses the challenges rising from this approach.

1 Introduction

The globalization of business and commerce makes enterprises increasingly dependent on their cooperation partners; competition takes place between supply chains and networks of enterprises. In this competition, the flexibility of enterprise information systems becomes critical. The IT systems and development teams should be able to respond in a timely way to the requirements arising from the changing co-operation networks and their communications needs.

The availability of open networks and the rise of service-oriented architectures have created an environment where collaboration between enterprise ICT systems becomes technically feasible. The current challenges on collaboration management focus on ensuring the semantics and pragmatics of collaborations.

It is especially interesting to capture the inter-enterprise business processes in such a way that autonomous computing systems can control and manage collaborations of that form. We need automated processes - automated within reason and trust - for creating inter-enterprise relationships so that the selected business processes can span this new, temporary business network. In these processes, fundamental tools are those that ensure interoperability in a technically and semantically heterogeneous environment. On the other hand, we need environments where new business process models can be developed, published, and evolved. Business applications, business needs, and business network topologies change rapidly, and that change has to be reflected by new business process models. Even more frequently there are pressures on changing the membership of an existing business network - a company fails, another provides a better quality service, yet another has more robust suppliers.

This paper describes how the foundations of RM-ODP (the reference model of open distributed processing) [8, 9] have been expanded and interpreted in the web-Pilarcos project for the benefit of open business network management. The goal of the project is to develop middleware services that support inter-organizational co-operation. The web-Pilarcos project aims at managing dynamic communities in a way where membership requires technical, semantic and pragmatic interoperability. The architecture design specifically addresses the needs of independent evolution of computing platforms, application services, and operational policies in each enterprise involved. The solution gives special emphasis to runtime expression of pragmatic aspects. The perspective taken is of the interoperability middleware developer. The concepts and services of the interoperability middleware become available for applications, both for service providers and service users.

In addition, this paper discusses some of the challenges rising from the need for new, global infrastructure services for B2B collaboration management. Some of these challenges can be addressed by the ODP development community itself, some others need to be resolved through industry-driven consortia.

Section 2 outlines the required functionality for establishing eCommunities, controlling their behaviour, and changing their behaviour and structure during their lifetime, drawing attention to the challenges for B2B middleware. The required concepts and their relationship to RM-ODP concepts are discussed in Section 3. Section 4 briefly describes a B2B middleware architecture and services that are partially addressed by RM-ODP functions but have been enhanced and refined by our work. The paper is concluded by challenges for future work.

2 Composing and controlling eCommunities

Our essential goal is to support dynamic collaboration between service components (even, enterprise applications), across autonomic enterprises. The basic idea is very close to the ones behind virtual enterprises (VEs) or extended enterprises, and builds on loosely-coupled, autonomous services.

Traditional extended enterprise models evolved from the intra-organizational integration of enterprise applications. The B2B application integration solutions lead to tight coupling of applications based on data-oriented integration, application-interface oriented integration, method-oriented integration, portal-oriented integration or process-integration oriented integration [22]. On the other hand, ERP systems were burdened with heavy development cycle overhead, as enterprise application changes, IT computing platform changes, and business process changes were not supported. The next wave of systems took up a more dynamic approach [31]. The second phase ERP systems allow dynamic configurations of applications with peer-to-peer relationships. Also, the business process control aspects and integration of workflow management have strengthened the area. Furthermore, distributed business process management systems have started to emerge.

A move from static, monolithic extended enterprises to dynamically managed, loosely connected VEs has taken place. However, the connection between VE management mechanisms and business process management is not yet well developed.

The challenges met with the loosely-coupled, open collaboration networks are three-fold.

1. The participating enterprises should be autonomous, and furthermore, the services becoming part of the collaboration network should be autonomically administered.
2. The B2B middleware should provide automatic facilities for ensuring interoperability within the managed collaboration networks.
3. The B2B middleware environment should provide a set of concepts for managing collaboration network membership, conditions, and dynamics. These concepts should be supported by pervasive middleware services.

Autonomy is one of the key design aspects. It spans

- selection of computing platform, and schedule of technical changes in it,
- selection of service components put externally available,
- evolution life-cycle of each offered enterprise application, including withdrawal of services already part of some VEs,
- decisions on the kind of collaborations that are entered,
- decisions on the kind of partners are accepted, and
- decisions on leaving existing collaborations.

Within each collaboration, situations may rise where the operational goals of the collaboration and an enterprise contradict. In contradictory situations, enterprises should be autonomic in deciding whether they act according to their internal interests (and expect the sanctions of contract breaches) or comply with the VE rules.

The autonomy challenge is addressed by the use of service oriented architectures. For example, Web Services technologies provide a suitable frame for hiding technical processing differences. The engineering and deployment detail of service provision is left for the enterprises to manage, and between enterprises, only such service features are made visible (as metainformation) that are relevant to interoperability and managing dynamic changes in communication.

Another essential autonomy requirement of enterprises is that they should be able to determine the set of potential partner enterprises, a set of trusted partners. Trust information services should become one of the global infrastructure services (compare: DNS name service is a global infrastructure service). Trust should be tagged to each resource, client, and VE, and the levels of trust be dependent on the socially and technically correct behaviour of the element, as seen by others in the network. Trust management is an integral part of a VE architecture.

Collaboration between service components or enterprise applications require interoperability. Interoperability – i.e. the effective capability for mutual communication of information, proposals and commitments, requests and results – requires technical, semantic and pragmatic interoperability [5]. Technical interoperability means that messages can be transported from one application to another, for example using a common transport protocol, or other shared signaling method. Semantic interoperation means that the message content is understood in the same way by the senders and the receivers. This may require transformations or other manipulation of messages, based on shared ontologies. Finally, pragmatic interoperability captures the willingness of partners for the actions necessary for the collaboration. The willingness to participate has two sides: capability of performing a requested action (for example, whether there is an application method available or not), and policy dictating whether the available action should or should not be performed (for example, a bank transfer handled after office hours).

The purpose of the B2B middleware is to provide a set of collaboration related concepts for application components to use, without the need of application software to include complex routines for example for partner discovery, interoperability guarantee, or change management. The concepts include community, role within the community, member of a community in a role, business process, policy, eCommunity contract, and contract breach, all of which have their counterparts in RM-ODP standards, as described in Section 3.

The most prevalent concept for our collaboration management architecture is the model of dynamic collaborations themselves, eCommunities. An eCommunity is controlled at operational time by an eCommunity contract. The eCommunity contract essentially uses a set of business process models as a model of the behaviour of members within the eCommunity. The collaborating services from each enterprise are aware of the business process model used between them, but do not implement the control of it. Instead, the control is left for B2B middleware services. These middleware services run metalevel protocols for controlling the eCommunity structure and state, including reports of contract breaches.

The life-cycle of an eCommunity has two modes:

- establishment phase supported by a breeding environment that ensures selection of appropriate partners and the interoperability of involved services, and
- operational phase supported by reflective control environment that manages dynamic changes in the eCommunity, and detects and resolves breaches of contracts.

The services of the breeding environment may be used even after reaching the operational phase, as the reflective control facilities may call upon restructuring of the eCommunity. The breeding environment should allow as open as possible route for enterprises to join in, giving an effective market for new partners. It provides facilities to a) populate an eCommunity to be created, b) negotiate eCommunity establishment, and c) commit eCommunity establishment.

The eCommunity management services at operational time are provided by the eCommunity contract object itself thought the following operations: a) terminate eCommunity, b) notify of entering compensation process, c) notify of detected eCommunity contract breach, d) query eCommunity contract metainformation and eCommunity status in terms of progress in the business process, membership, and breach management process definitions, e) repopulate and negotiate an existing eCommunity; and f) for members to join/leave an eCommunity role.

Access to these operations is made available at each platform at each administrative domain, regardless of whether the service is actually realized locally or remotely supported. (Different deployment models even open up new electronic service market opportunities.)

The middleware services providing for these services are discussed further in Section 4.

3 Refinement of RM-ODP concepts

The ODP standards provide for system architectures that allow distributed information processing applications to collaborate in a heterogeneous environment and under multiple organizational domains [7]. The ODP standards direct systems to be built so that they support cost-effective interoperability of applications, despite their implementation using different platform architectures and resources. For this, it is essential to accommodate system evolution and runtime changes, and define a transparency support framework for communication.

The RM-ODP provides a division of an ODP system specification into viewpoints, in order to simplify the description of complex systems [9]. The viewpoint languages each refine a set of general concepts defined for the reference model [8] The enterprise viewpoint language has been further developed [16], as it brings in business related aspects. Furthermore, the reference model defines structuring rules and functions for a supporting infrastructure for global computing. The functions that have been further standardized include the trading service [10], naming framework [12], type repository function [15], and interface binding framework [11] together with the supporting protocols [14].

In the web-Pilarcos architecture, we have tackled the challenges of autonomy, interoperability, and suitable concepts and services for managing eCommunities with these tools. Conceptually, the terms service and service offer management, community, federation and contract deserve further attention.

The concept of service is missing from the RM-ODP model, although it is mentioned in various term explanations (e.g. object [8, 8.1]). We define *service* as 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).

Services are provided by administrative domains, for example by enterprises, departments or any independent ICT systems. The administrative domains are the units of autonomy within our model. How engineering and deployment of services are organized within the administrative domain is hidden from the service users and B2B management services. Only management functions within the administrative domain, like node or object management, are involved with the technology and engineering of these.

In RM-ODP, models of behaviour (including interactions between objects and internal actions) are restricted to signals, announcements and interrogations, described using interface signatures. Signatures reveal operation names and data types, as well as parameters involved. However, for business processes, more complicated choreographies need to be expressed, although built on top of these basic primitives. In addition, these primitive actions need to be attached with nonfunctional features too, like QoS or trust.

For these challenges, enhancing RM-ODP with more elaborate conversation models is not the right solution. Instead, tools for introducing and reflecting such models are needed. Such models are specific to application areas, and may need to evolve in time, or may be negotiable between collaboration partners. In contrast to this, association of nonfunctional features to interfaces should become an integral part of the RM-ODP model, but again leaving the ontologies of features and usable values open for extention.

For the use in eCommunities, 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 structuring rules of the web-Pilarcos style of service offers is captured in Figure 1. The structure is more detailed than can be found in the ODP trading function standard, which 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. [30]) by not addressing the groundings (access details) or the actual location of services. These locating aspects are only captured by the eCommunity contract formed according the offers; the grounding aspects are considered private for each enterprise. Only the interoperability-related features are required.

By capturing all viewpoint aspects into the service offer, we address the interoperability ensuring challenge. Making metainformation available in such structured way, we use the interface matching mechanisms of the trading service to match all relevant aspects of interoperability at the same time. This of course applies only for static analysis; for example for policies subject to further changes, only dynamic monitoring can catch mismatches.

This kind of matching process requires that the service offers are expressed in commonly understood terms in the areas of business processes and services within that context, nonfunctional features associable to processes or services, and policy frameworks meaningful for the services in question. In this area, some ontology creation tools exist, but there is no consensus on what principle the ontologies should be organized on.

```
service offer        := ((interface syntax)
                         (interface protocol)
                         (information el format)
                         (nonfunct aspects)*)*
                         (policies)
                         (platform requirements)
                         (channel requirements)
interface syntax     := <IDL specification> |
                         <WSDL specification>
interface protocol   := <partial ordering
                         rules of operations
                         in syntax>
nonfunctional aspects:= <QoS offer> |
                         <trust requirement> |
                         <security mechanism name>
information element format
                     := <schema>
policies             := <policy framework
                         name> <policy name>
                         <policy value offer>
platform requirement := <platform name>
channel requirements := <channel type name>
                         <binding type name>
```

Figure 1. Structure of service offers.

The concept of *community* is used for describing the collaboration of several services. The ODP enterprise language specifies a community as a configuration of enterprise objects with a contract on their collective behaviour [16, 5.1.1]. The community specification includes [16, 5.2]

- a set of roles; the role specification gives requirements and restrictions for the behaviour of an object;
- rules for assigning enterprise objects to roles; the policy rules can address individual objects or relationships between objects, and can make restrictions on behavioural and non-behavioural properties of the potential objects;
- policies that apply to roles; policy values act as selectors for alternative behaviours for the objects – and thus also for the community;
- description of behaviour that changes the structure or the members of the community during its lifetime.

Each role [9, 9.14] in the community specification denotes a possible behaviour. The behaviour descriptions are refined with policy statements indicating which parts of the behaviour are prohibited, permitted or obliged to take place and under what conditions.

A role can be populated by an object that represents another community. In this way, larger systems can be composed of subservices. Functional composition is better supported by inclusion of multiple community specifications into a system specification and definition of the relationships between communities.

A community specification may be divided into several epochs, each epoch [9, 10.5] presenting a different set of services supported by the community. For instance, a service might have a configuration phase and an operational phase; during the configuration phase only a management interface is available, but during the operational phase the actual service interfaces are also available.

The ODP community structure is used as a baseline for defining eCommunity contract structure in web-Pilarcos. The eCommunity contract has the structure that is outlined in Figure 2.

The eCommunity contract structure is determined by the selected business network model. This model is suggested by the initiator of the eCommunity establishment. The models need to be available through a shared repository, so all potential partners can assess whether the goal, structure and terms of the community are acceptable.

Besides roles to be populated by services, the template shows requirements for the binding objects that are needed for realizing interactions between roles. For bindings, we expect an explicit, open binding object [13, 1]. The binding object provides a framework and interface for the communication service. The binding object also provides a management interface through with the internal structures can be configured using the local communication facilities or additional helper components. The benefit of this model is that the requirements on shared platform services become minimal: We need common understanding of interface descriptions and common understanding of a few alternative communication channel structures.

In addition to the metainformation contents, the contract object provides operations for changing members and community structure.

For the purposes of web-Pilarcos architecture, we have connected the role behaviour to a service behaviour. Thus, assignment rules are directly related to import requests from the trading service for suitable service offers from potential members of the eCommunity. This is in line with current trend of service oriented architecture (SOA) where the definition of abstract service, service discovery, and semantic composition of services are topical [25, 24, 24].

Service oriented architecture, and more specifically, web services provide evidence for the industrial movement towards independently developed and administered services.

- *reference to the business network model;*
- *current epoch information;*
- *process for changing epoch;*
- *for each role*
  - *assignment rules that specify the requirements on*
    - *service type;*
    - *nonfunctional aspects;*
    - *restrictions on identity, participation on other eCommunities, etc;*
  - *conformance rules that are used for determining conformance to the role which the assigned component is in the role; similar as above;*
- *for each interaction relationship between roles*
  - *channel requirements*
  - *locations of the channel endpoints*
  - *QoS agreement*
  - *security agreement*
  - *information presentation formats*
- *for each policy that governs the choices between alternative behaviour patterns in the business network model*
  - *acceptable values or value ranges;*
- *references to alternative breach recovery processes;*
- *objective of the eCommunity (rules that can be used for various nondeterministic choices in the eCommunity; for example, what kind of attributes are more attractive when selecting a new member, available info for these rules is in service offers and in the business network model and in policy values of this eCommunity)*

Figure 2. eCommunity contract information.

The service itself has become the key element: the behaviour pattern, nonfunctional features, and contracting for providing a specified service in a context. The context is defined by an environment, including local and network resources, availability of required services, etc.

As the business network model captures services in terms of only their interface and external exchanges of information, the model is free from private information flows and workflows within the service providing organization or unit. This is a great benefit, as many industrial approaches on inter-enterprise workflow and business process modeling have reported that the current modeling languages and tools enforce too tight coupling between partners [4, 27]. The phenomenon is a natural consequence of the development history through integrated ERP systems, to A2A integration, and further to process-aware B2B integration. The VE approaches however do not yet have sufficient support for business process management.

The basic ODP concepts introduce communities and contracts as a way of expressing how the partners can reach a shared goal. However, no notation or further refinement has been given for expressing the goal or behaviour. We have chosen to use an ad-hoc enterprise viewpoint language for defining these aspects. The language resembles XML-based business process modeling languages and workflow languages; in practice, the service descriptions use enhanced WSDL descriptions [26].

We have not adopted the UML notations nor taken MDA [28, 3] as a driving force for the design. As discussed above, the focus in this work is not in the generation of service implementations themselves, but in composing eCommunities from existing services. We do not use the term reuse here, as the facilities for establishing collaborations is the primary goal. Naturally, the descriptions required by our infrastructure and those of MDA tools overlap, and this is to be considered as a great benefit. However, there is a difference between the bias towards ergonomic modeling tool view of the business network and the bias towards management software beneath.

It should be noted that with this work, we do not drive the standardization of domain specific business processes in itself, but standardization of facilities that help in evolution. A methodology where new standard processes or new suggestions can be published and adopted efficiently is a more persistent approach.

The ODP interface references and bindings [11] standard discusses management of *explicit binding objects*. Introduction of such binding objects with a set of selective transparency support and nonfunctional aspects management is needed. Many commercially interesting consortia recommendations on the area of inter-enterprise workflows, web services choreographies, and business process management systems expect a transaction-aware communication layer to appear. Although the current ODP standards create a placeholder for a unifying structure, it is not concrete enough to guide the isolated development trends for cross-platform protocols and services in this area.

The RM-ODP defines *<X> federation* as a community of *<X> domains* where there is a shared objective [9, 5.1.2,5.1.1]. A <X> domain is defined as a set of objects with a shared controlling object over the characteristic feature X [9, 10.3].

In the web-Pilarcos architecture, we form federations between eCommunity management agents in administrative domains. An administrative domain can be seen for example as an enterprise, a division, or another unit: essentially the domain is the unit of autonomy within our model. The objective of the federation of eCommunity management agents is collectively to form, maintain and use application level services from their domains in the roles of the eCommunity. As helpers, each agent has local management services, such as node or object management, monitors that are able to report breaches from the assumed interaction pattern, and binding factories. The shared goal of the management agent federation is to keep the agreed eCommunity running according to the contract.

4 Open B2B infrastructure services for collaboration management

In the web-Pilarcos environment middleware services for B2B collaboration management fall into two categories: cooperative management services for multidomain applications and local element management services [20].

The breeding environment services include only cooperative services:

- The standard trading service [23] for maintaining a repository of service offers, for the use of the enhanced trader.
- The enhanced version of ODP type repository [6] for holding relationship information between generic types (service types, binding types, interface types) that are technology-independent and used for matching purposes and technology-dependent templates that are used for instantiating the corresponding components and objects [17]. This mapping information is created by system programmers separately from business architecture descriptions and service offers.
- The repository for publishing and relating various business process models.
- The enhanced trader for populating business architectures with selected components [21]. The business process models contains roles as placeholders for services, but the selection of services for neighbouring roles is not independent, due to, for example, the need for shared binding requirements.
- The federation manager for negotiating, maintaining and renegotiating the eCommunity contract that represents an application instance.

The cooperative management services – enhanced trading, trading, type management, eCommunity management, federated binding – are all services that have a local server running in each domain. These active agents take care of making requests to their peers in other domains, as there is otherwise no authority to invoke management actions in a foreign domain [18]. The requests carry contracts to pass relevant meta-information that identifies what should be done and how.

The operational environment services include both local and cooperative services. The most essential cooperative service is that of the eCommunity contract object itself, with its management operations. The interface to the replicated contract object is maintained by all eCommunity managers.

In traditional protocol systems, interoperability could be verified statically, although with practical limitations and with expense. However, the process models described above cannot be fully verified any more. Two aspects, the deontic guards on actions and pragmatic monitoring of resources cause a situation where only some of the joint behaviour can be verified. For example, we can determine non-interoperability if available functionality is not sufficient for safe communication, leaving guards and pragmatic decisions aside. Or, non-interoperability can be stated if guards on actions are so contradictory that there are no acceptable traces through the process left.

Therefore, we need to add dynamic verification into the system, meaning introduction of

- monitors that enforce enterprise policies on resources (pragma) and notifies about discrepancies caused,
- monitoring of external service interaction conformance to the business process, and notification if inconsistent actions occur, and
- controllers of channels and notifications when channel properties have changed in a significant way.

The local service management adds lifecycle services and local bindings to this list.

- Service deployer for instantiating components for each role according to the contract that represents the application instance. The service deployer uses type repository information to map the contract onto appropriate technology solutions [21].
- Binding factory for instantiating communication channels between components. Because no remote instantiation service is supported across organizational boundaries, the binding factories at each computing system involved must cooperate. Again, the factories use repositories for mapping contract information onto appropriate engineering solutions [21, 19].
- Implementation repository for storing software packaging and maintaining their automatic installation scripts.

The overall view to the operational environment is twofold: First of all, the service components interact successfully with their peers in the eCommunity through binding objects. We call this the real system. Secondly, there is a level with a set of protocols for monitoring, configuring and reorganizing the real system constantly. We call this the metalevel. The relationship between these two layers is taken from reflective system design.

In reflective systems [2], the metainformation constantly describes the current real system structure, topology, state, qualities, etc. The essential part of the system infrastructure services are those facilities that are needed to keep the real system and the metadata in causal connection with each other. This means that changes in metainformation need to cause changes in the real system, and vice versa. For example, if an eCommunity member fails permanently, the eCommunity contract object reports that the member has abruptly left the eCommunity. Furthermore, the eCommunity contract object is proactive and starts the search for a replacing member in the eCommunity. After commitments from other members are received, the new member is joined to the eCommunity contract, and consequently, the service component is started up and bound to its peers through binding objects. Changes of metainformation can also be such that they only take effect later in the real system. For example, change in an enterprise policy is not necessarily effective immediately.

5 Conclusion

The above discussion shows that the RM-ODP concepts are suitable for modeling enhanced system software services for automated management of dynamic eCommunities. However, the standard definitions give fairly vague direction to the work.

In the design, we have noted the need for further ODP functions, such as

- business process model repository;
- community aware coordination functions; and
- trust management functions (build on top of security functions).

Interoperability support mechanisms require definition of new ontologies for defining terms and semantics for elements in the business or service models, and interoperability related attribute sets to be used together with service types. Similarly, ontologies for policy sets relevant for business processes and resources would be needed.

Furthermore, some existing concepts and framework standards need refinement. For example, concepts related to service and various alternatives for explicit binding object architectures were discussed above.

6 Acknowledgments

This article is based on work performed in the Pilarcos and web-Pilarcos projects at the Department of Computer Science at the University of Helsinki. The Pilarcos project was funded by the National Technology Agency TEKES in

Finland, Nokia, SysOpen and Tellabs. In web-Pilarcos, active partners have been VTT, Elisa and SysOpen. The web-Pilarcos project is a member in national ELO program (E-Business Logistics) [29]. The work is strongly integrated with RM-ODP standards work, and recently has found an interesting context in the FP6 INTEROP NoE collaboration.

References

[1] G. S. Blair, G. Coulson, N. Davies, P. Robin, and T. Fritzpatric. Adaptive Middleware for Mobile Multimedia Applications. In *Proceedings of the 8th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, 1997.

[2] G. Coulson. What is reflective middleware? *IEEE Distributed Systems Online*, 2003. Area on Reflective Middleware – http://dsonline.computer.org/middleware/RMaraticle1.htm.

[3] D. S. Frankel. *Model Driven Architecture - Applying MDA to Enterprise Computing*. OMG Press, 2003.

[4] D. Hollingsworth. *The Workflow Reference Model: 10 Years On*. Fujitsu Services, UK; Technical Committee Chair of WfMC, 2004.

[5] INTEROP NoE, EU FP6. Interoperability research for networked enterprises applications and software. http://interop.aquitaine-valley.fr/.

[6] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Type Repository Function*. IS14746.

[7] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 1: Overview*, 1996. IS10746-1.

[8] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 2: Foundations*, 1996. IS10746-2.

[9] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Part 3: Architecture*, 1996. IS10746-3.

[10] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. ODP Trading function. Part 1: Specification*, 1997. IS13235-1.

[11] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – ODP Interface References and Binding*, Jan. 1998. IS14753.

[12] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – ODP Naming framework*, 1998. IS14771.

[13] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. Interface references and binding*, 1998. IS14753.

[14] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing – Protocol Support for Computational Interactions*, 1999.

[15] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. Reference Model of Open Distributed Processing. ODP Type repository function*, 1999. IS14746.

[16] ISO/IEC JTC1. *Information Technology – Open Systems Interconnection, Data Management and Open Distributed Processing. ODP Enterprise Language*, 2003. IS13235.

[17] P. Kähkipuro, L. Marttinen, and L. Kutvonen. Reaching Interoperability through ODP type framework. In *TINA'96 Conference: The Convergence of Telecommunications and Distributed Computing Technologies*, pages 283 – 284. VDE Verlag, Aug. 1996. Extended abstract.

[18] L. Kutvonen. Management of Application Federations. In H. Konig, K. Geihs, and T. Preuss, editors, *International IFIP Working Conference on Distributed Applications and Interoperable Systems (DAIS'97)*, pages 33 – 46, Cottbus, Germany, Sept. 1997. Chapmann & Hall.

[19] L. Kutvonen. *Trading services in open distributed environments*. PhD thesis, Department of Computer Science, University of Helsinki, 1998.

[20] L. Kutvonen. Automated management of interorganisational applciations. In *EDOC2002*, 2002.

[21] L. Kutvonen, J. Haataja, E. Silfver, and M. Vähäaho. Pilarcos architecture. Technical report, Department of Computer Science, University of Helsinki, Mar. 2001. C-2001-10.

[22] D. S. Linthicum. *B2B Application Integration - eBusiness-Enable Your Enterprise*. 2001.

[23] Object Management Group. *OMG Trading Object Service Specification*, June 2000. OMG formal/2000-06-27.

[24] M. P. Papazoglou and D. Georgakopoulos. Service oriented computing. *Commun. ACM*, Oct. 2003.

[25] M. P. Papazouglou and W.-J. van den Heuvel. Service-oriented computing: State-of-the-art and open research issues.

[26] T. Ruokolainen. Component interoperability. Master's thesis, University of Helsinki, Department of Computer Science, 2004. In Finnish.

[27] K. Schulz, K.-D. Platte, T. Leidig, R. Guggaver, K. Elams, A. Zwegers, F. Lillehagen, G. Doumeingts, A. Berre, M. Anastasiou, M. Nunez, R. Goncalves, D. Chen, and M. Missikoff. A gap analyisis – interoperabilty development for enterprise application and software - road maps. Technical report, 2003.

[28] J. Siegel. *Developing in OMG's Model-Driven Architecture*. Object Management Group, Nov. 2001. White paper, revision 2.6.

[29] TEKES. *ELO program*, 2003. http://www.tekes.fi/programs/elo.

[30] The Intelligent Software Agents Group The Robotics Institute Carnegie Mellon University. Semanatic matchmaking for web services discovery. Technical report, 2003. http://www.damlsmm.ri.cmu.edu/.

[31] M. Ulieru and R. Unland. Emergent holonic enterprises: How to efficiently find and decide on good partners. *International Journal of Information Technology and Decision Making*, 2(4), Dec. 2003.